QEMM

Quarterdeck

Version 7.0

Quarterdeck Expanded Memory Manager

386, 486 processors and Added Support for Intel Pentium, DOS v6.00 and Windows v3.10.

Reference Manual

COPYRIGHT

1901 Main Street, Santa Monica, CA 90405 • (310) 392-9851

U.S. Patent Number 5,237,669

TRADEMARKS

License

United States:

This License is your proof of license.
Please treat it as valuable property.

QUARTERDECK END USER LICENSE AGREEMENT (THE "AGREEMENT")

NOTICE TO END USER: CAREFULLY READ THE FOLLOWING LEGAL AGREEMENT. USE OF THE SOFTWARE (THE "SOFTWARE") (AND FONTS, IF ANY) PROVIDED WITH THIS AGREEMENT CONSTITUTES YOUR ACCEPTANCE OF THESE TERMS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, PROMPTLY RETURN THE SOFTWARE AND THE ACCOMPANYING ITEMS (INCLUDING WRITTEN MATERIALS, BINDERS AND CONTAINERS) TO THE LOCATION WHERE YOU OBTAINED THEM FOR A FULL REFUND.

1. License Grant. Quarterdeck Office Systems, Inc. ("Quarterdeck") grants to you (either as an individual or entity) a nonexclusive sublicense subject to the provisions hereof: (a) to use the SOFTWARE solely for your own internal personal or business purposes on a single computer (whether a standard computer or a workstation component of a multi-user network). You may not copy the written materials accompanying the SOFTWARE. This Agreement is effective until the year 2040.

2. Proprietary Rights. You acknowledge that the SOFTWARE is proprietary to Quarterdeck and its suppliers. You agree to hold the SOFTWARE in confidence, disclosing the SOFTWARE only to authorized employees having a need to use the SOFTWARE as permitted by this Agreement and to take all reasonable precautions to prevent disclosure to other parties.

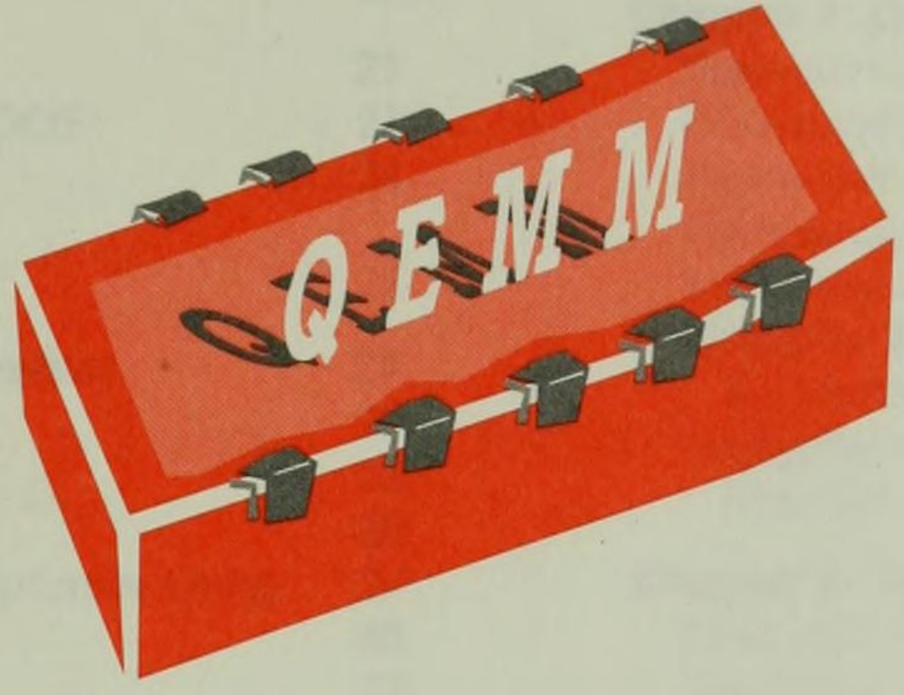
3. Other Copies. You will not make or have made, or permit to be made, any copies of the SOFTWARE or portions thereof, except as necessary for its use with a single licensed computer system under the terms and conditions of this Agreement. You agree that any such copies shall contain the same proprietary notices which appear on or in the SOFTWARE.

4. Ownership. Except as stated above, this Agreement does not grant you any rights to patents, copyrights, trade secrets, trade names, trademarks (whether registered or unregistered), or any other rights, franchises, or licenses in respect of the SOFTWARE. Title to and ownership of the SOFTWARE, any reproductions and any documentation shall remain with Quarterdeck and its suppliers. You will not adapt or use any trademark or trade name which is likely to be similar to or confusing with that of Quarterdeck or any of its suppliers or take any other action which impairs or reduces the trademark rights of Quarterdeck or its suppliers.

(License agreement continued on rear inside cover.)

QEMM

Quarterdeck



500US-QM3750

Reference Manual

Credits

The QEMM program was written by Dan Spear, Larry Mayer, Phil Gardner, Steve Goodrich, Mike Wacker, Mark Indictor, Brian Breidenbach, Russell Bonakdar, Russell Bell, Jim Susoy, Steve Sprigg, David Bromberg, Dan Travison, Larry Rush and Todd Threadgill.

The manuals were written by Phil Glosserman and Dan Sallitt.
The layout is by Phil Glosserman, Yvonne Johnson and Laura Camuti.
Proofreading and editing by Laura Camuti.

Table of Contents

1 Introduction

Thank you for purchasing Quarterdeck's Expanded Memory Manager—QEMM. QEMM works behind the scenes to provide you with as much memory as possible for running programs, thus freeing you from the details of memory management. Since 1985, Quarterdeck has been the leader in bringing memory management innovations to DOS, and for that reason, QEMM has consistently garnered top awards for memory management software in numerous computer trade publications including PC Magazine, InfoWorld, Byte and PC Computing.

Among numerous other enhancements, the new version of QEMM features both a DOS and a Microsoft Windows interface—you can use whichever you prefer.

You do not have to be a computer expert or understand memory management to use QEMM. Most users will just perform a simple automated installation and immediately enjoy the benefits of using QEMM. If you are eager to start, skip to the *QEMM Installation Guide* now.

What is so important about QEMM? Not only do today's computers have more power and memory, but PC programs are larger and make greater demands on memory than ever before. A typical PC may be simultaneously running a multitasking environment such as DESQview or Microsoft Windows, a disk compression utility such as Stacker or DoubleSpace, multiple TSRs (memory-resident programs, such as DOS 6's VSafe anti-virus program) and device drivers for a network, mouse, CD ROM, sound board, and other peripheral devices. And, different programs use different kinds of memory: conventional memory, upper memory, expanded memory, and extended memory. QEMM manages all these kinds of memory and can automatically transform your PC's memory into whatever kind of memory your program needs. QEMM is compatible with MS-DOS, IBM DOS, DR DOS and Novell DOS, and supports DOS, DOS Extended and Microsoft Windows applications.

QEMM uses advanced techniques to create the maximum amount of first-megabyte memory for running programs on your system. QEMM allows you to run larger applications, by giving your

programs as much conventional memory as possible. QEMM even helps your applications to run faster. Plus, QEMM includes several features that can provide additional memory, over and above that provided by MS-DOS 6's memory manager.

QEMM is solid and dependable—time-proven by millions of users. Whether you are using DOS version 3, 4, 5 or 6, QEMM will provide you with superior, hands-free management.

In addition, when combined with Quarterdeck's DESQview, QEMM turns DESQview into DESQview 386—a powerful 386 multitasking control program.

PC memory management is a technical subject, but you need not understand it to use QEMM. QEMM manages your PC's memory behind the scenes—most users will only need to perform a simple installation, then forget about memory management altogether. If you are an advanced user interested in the details of memory management, QEMM has advanced features and parameters you can use to fine-tune memory management on your PC.

If you are upgrading from an earlier version of QEMM, the new version of QEMM has several enhancements, including support for Intel's Pentium processor, the ability to load DOS in High RAM and the ability to free up more RAM by special management of MS-DOS 6's DoubleSpace and DriveSpace device drivers. As mentioned earlier, this version of QEMM includes an optional Microsoft Windows interface. Many of QEMM's improvements are "under the hood"—QEMM is faster, more efficient and can free up even more memory for your applications.

Where to Go From Here

The rest of this chapter gives some basic information about PC memory and describes QEMM's features. If you are eager to get started, you can skip to the *QEMM Installation Guide*. As mentioned earlier, most users will only need to perform a simple installation, then leave the details of memory management to QEMM.

QEMM is for IBM and IBM-compatible PCs and PS/2s equipped with 386SX, 386DX, 386SL, i486DX, i486SX, or Pentium processors. In this guide, we will use the terms *386* and *80386* inclusively, to refer to all of these PCs.

The Basic Types of PC Memory

Here is a brief discussion of the different kinds of memory available on 386 PCs.

- **Conventional memory** starts at 0K and normally ends at 640K; it is where most DOS programs run. Most DOS programs run in *real mode* (the main processor's basic operating mode), which only allows access to the first megabyte of memory addresses (0-1024K). Because the addresses between 640K and 1024K are reserved for ROMs and system hardware, real mode programs must normally run in the area below 640K. Without the aid of a memory manager like QEMM, most TSRs (memory-resident programs) and device drivers also occupy conventional memory, reducing the amount of memory available to run DOS programs.

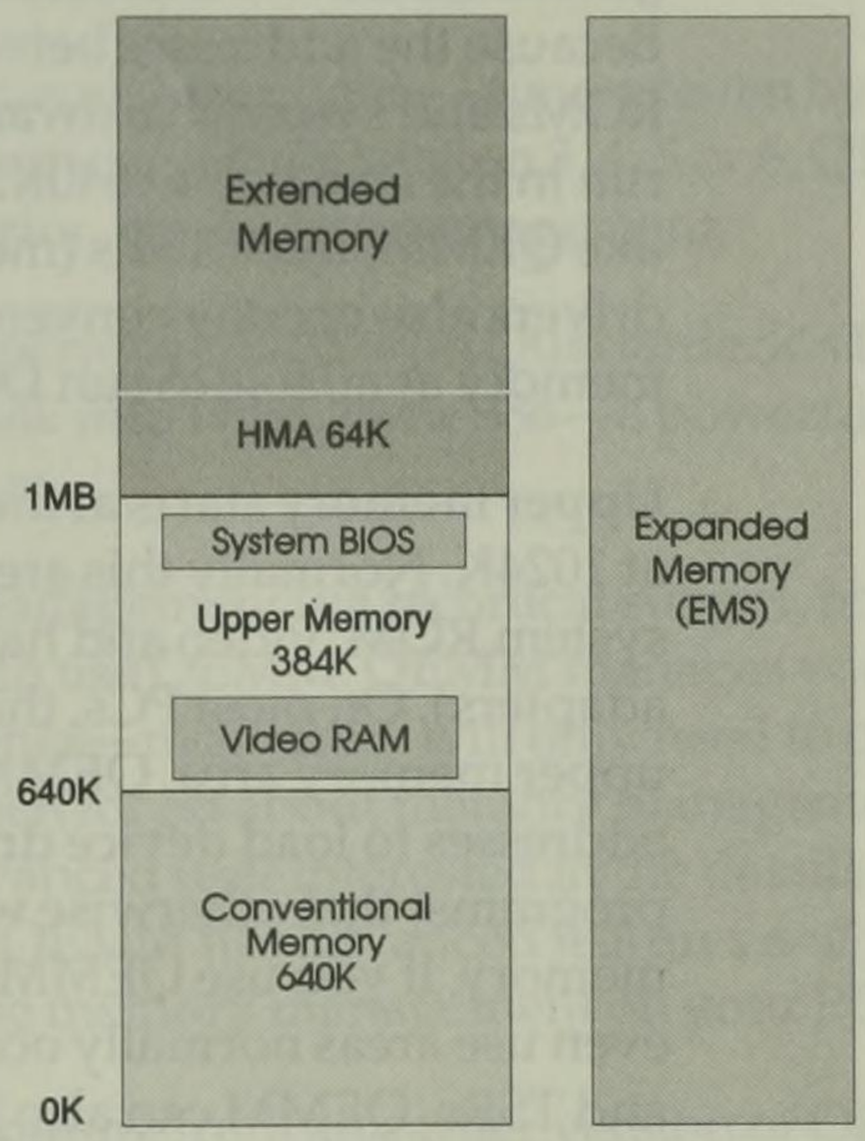
- **Upper memory** starts at the end of conventional memory and ends at 1024K. Normally this area is set aside for use by hardware—the system ROM, video and hardware adapter cards (e.g., network adapters). On most PCs, this hardware does not use the entire upper memory area. QEMM can use the unused upper memory addresses to load device drivers and TSRs (memory-resident programs) that otherwise would occupy space in conventional memory. If you use QEMM's Stealth ROM feature, QEMM can even use areas normally occupied by ROMs to load device drivers and TSRs. QEMM can also load selected parts of DOS into the upper memory area. For information on loading items into the upper memory area, see "What Is High RAM?" on *page 9*.

- **Extended memory** is the memory addressed above 1024K (i.e., memory above the first megabyte of memory addresses)—it is used by programs operating in protected mode. *Protected mode* is a collective name given to several advanced operating modes of 80286 and higher processors. Programs that run in protected mode differ from real mode programs in that they can access memory above the first megabyte of memory. Protected mode is used mainly by DOS-extended programs, XMS programs, memory managers such as QEMM, and multitasking control programs such as Microsoft Windows.

  A DOS-extended program is a special DOS program that runs in your PC's protected mode, but switches back to real mode when it needs to use DOS or the BIOS. AutoCAD 12, Lotus 1-2-3 Release 3, Oracle Professional, IBM Interleaf and Microsoft Windows 3.0 and 3.1 (in standard or enhanced mode) are DOS-extended programs.

- **High memory area (HMA)** is the first 64K of extended memory. The HMA differs from the rest of extended memory because the 286 and higher processors can access it in real mode. Some

programs (e.g., DOS, DESQview) can store a portion of their code in the HMA, freeing up conventional memory which would otherwise be used by that code.

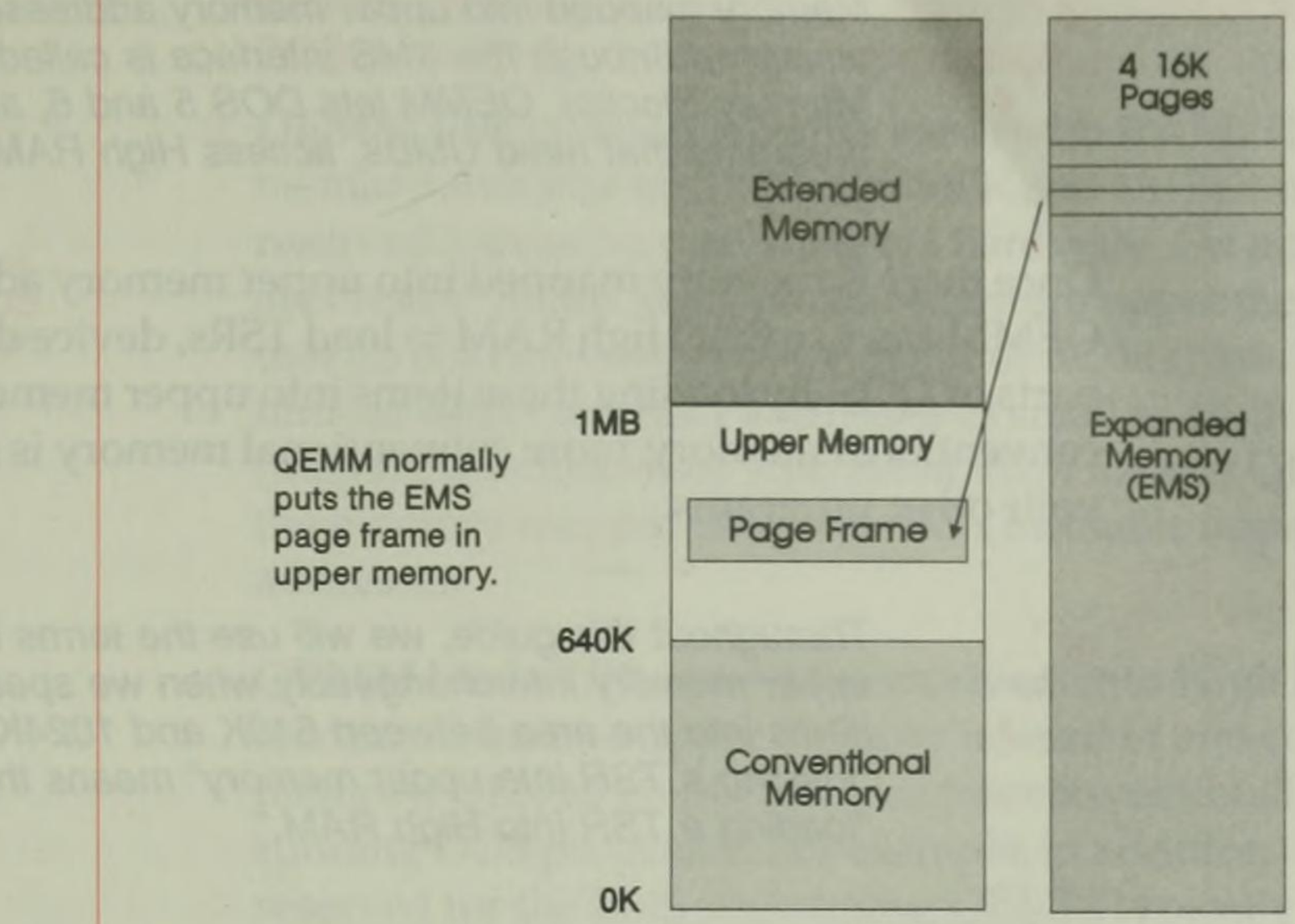


The basic kinds of PC memory

The standard uses of extended memory are defined in formal specifications called XMS, VCPI and DPMI. For information on these specifications, see **Appendix C**.

- **Expanded memory (also called EMS)** is memory outside the PC's memory address space that an expanded memory manager such as QEMM can cause to appear within the first megabyte so it becomes accessible by DOS real mode programs. Some DOS programs use expanded memory to store and access data that will not fit in conventional memory. In most cases, a program that uses EMS accesses up to 64K of expanded memory at a time (in 16K units called pages) at a special area of upper memory called the *page frame*. Originally, on PC models lower than 386s, EMS was memory you added onto your PC with an EMS adapter card. QEMM uses the 386 processor's mapping capabilities to transform your PC's extended memory into expanded memory for programs that request this type of memory.

It is important to distinguish between extended memory and expanded memory. You may think of extended memory as the



QEMM normally puts the EMS page frame in upper memory.

Up to four 16K memory pages from outside the first megabyte of memory are made to appear as if they are within the first megabyte.

Expanded memory (EMS)

physical memory above one megabyte. Expanded memory is memory outside the PC's memory address space that an expanded memory manager program such as QEMM maps into the first megabyte. *Mapping* is the process of making a portion of memory (on 386 PCs, usually extended memory) appear at a different address. In general, extended memory was developed in anticipation of protected mode programs, and expanded memory was developed to increase the power of real mode programs.

What is High RAM?

The upper memory area (the area between 640K and 1024K) is normally reserved for use by your system's hardware. On many PCs, there are "holes" in this area—that is, there are upper memory addresses that do not have any physical RAM chips. QEMM uses mapping to make RAM "appear" in those areas. By mapping memory, QEMM can "fill out" any missing physical memory in the first megabyte of memory. *High RAM* is QEMM's name for the memory mapped into the upper memory addresses that are not used by ROMs or adapter RAM. QEMM can fill areas as small as 4K with High RAM.

Memory mapped into upper memory addresses that can be accessed through the XMS interface is called UMBs (Upper Memory Blocks). QEMM lets DOS 5 and 6, and other programs that need UMBs, access High RAM as UMBs.

Once there is memory mapped into upper memory addresses, QEMM can use that High RAM to load TSRs, device drivers, and parts of DOS. By loading these items into upper memory instead of in conventional memory, more conventional memory is available for your other programs.

Throughout this guide, we will use the terms High RAM and upper memory interchangeably when we speak of loading items into the area between 640K and 1024K. That is, "loading a TSR into upper memory" means the same thing as "loading a TSR into High RAM."

QEMM's Benefits and Features

QEMM offers the following benefits and features:

Benefit: QEMM sets up your PC's memory optimally—without you having to know anything about memory.

- **QEMM's Express Install automatically examines your system and determines how best to configure its memory and makes any necessary changes to your system's CONFIG.SYS and AUTOEXEC.BAT files—requiring no input from you.** You can install QEMM and optimize your system in under five minutes.

- **QEMM responds to your programs' requests for memory and provides them with the kind of memory they need (expanded or extended)—with no need for intervention by you.**

- **Manifest, Quarterdeck's system reporting program, gives you a detailed description of your system and its memory**—valuable if you want over-the-phone assistance in understanding your system. QEMM includes both DOS and Windows versions of Manifest.

√ indicates a new or enhanced feature.

+ indicates a feature not found in MS-DOS 6's memory manager.

√ indicates a new or enhanced feature.

+ indicates a feature not found in MS-DOS 6's memory manager.

Benefit: QEMM gives you the maximum amount of free conventional DOS memory for running your programs.

- **QEMM fills missing memory areas with RAM.** The upper memory area (the area between 640K and 1MB) is normally reserved for use by your system's hardware. On many PCs, there are "holes" in this area—that is, there are upper memory addresses that do not have any physical RAM or ROM chips. QEMM finds unused upper memory addresses as small as 4K in size and fills (maps) these addresses with RAM. *High RAM* is QEMM's name for the memory mapped into your PC's available upper memory addresses.

- **QEMM loads TSRs and device drivers into High RAM.** By loading items into upper memory instead of into conventional memory, QEMM gives you more free conventional memory for running DOS programs. For example, in addition to the 64K reserved for the EMS page frame, QEMM typically finds 128K of usable upper memory on IBM PS/2s and 128K-160K on other PCs. And, on most PCs, QEMM's Stealth ROM feature (see below) can find significantly more usable upper memory. When QEMM loads an item into High RAM, we say the item is "loaded high."

+ - **DOS-Up can load parts of DOS into High RAM.** QEMM's DOS-Up feature can load DOS's command processor, the DOS kernel, DOS Data, and DOS resources (FILES, BUFFERS, STACKS, LASTDRIVE and FCBS) into High RAM. If you have DOS 5 or 6, this feature can typically free up 7K-70K of conventional memory, depending on how your system is configured. If you have DOS 3 or 4, this feature can free up 50 to 70K of conventional memory.

- **Optimize automatically configures upper memory.** QEMM's Optimize program examines the device drivers and TSRs (i.e., memory resident programs) that are loaded at system startup time and determines which ones can be moved from conventional to upper memory. Optimize analyzes all possible ways of loading them, then moves them to the locations that ensure the best fit. By moving these items to upper memory, Optimize frees up more conventional memory to run your programs. Optimize is the reason you do not have to be a PC expert to make the best use of your PC's memory.

√ indicates a new or enhanced feature.

+ indicates a feature not found in MS-DOS 6's memory manager.

+√ ❑ **Optimize lets you restore previous configurations.** Whenever you run Optimize, it creates a backup of your previous configuration. The installation process also creates a backup of the configuration that existed before QEMM. Thus, you can easily restore any of your nine most recent configurations or the configuration that existed before you installed this version of QEMM. In the unlikely event of a conflict, you can always restore a configuration that you know worked. This feature makes installing QEMM and optimizing memory fail-safe processes.

+ ❑ **Squeeze finds extra space for TSRs and device drivers to initialize.** Often, TSRs and device drivers require more memory to initialize than they do once they are resident. Optimize has a feature called Squeeze that ensures that such programs get the extra memory they need to initialize. The Squeeze feature relinquishes this extra memory when the program finishes initialization. Without Squeeze, certain items might not load high because their initialization sizes are larger than any contiguous available upper memory area.

+√ ❑ **QEMM's patented Stealth ROM feature uses the EMS page frame to manage your PC's ROMs, freeing up an additional 48K-115K of upper memory ROM addresses for use by TSRs and device drivers.** When Stealth ROM is enabled, QEMM can create up to 224K of High RAM on IBM PS/2s, Compaqs, and many other PC compatibles. QEMM can use about 16K of High RAM to load itself. In the new version of QEMM, Stealth ROM has been enhanced to be automatically compatible with more machines.

+√ ❑ **QEMM's Stealth D*Space feature (pronounced "Stealth Dee-space") uses the EMS page frame to manage MS-DOS 6's DoubleSpace or DriveSpace device driver, freeing 31K to 49K of memory, depending on your configuration.** (This feature was formerly called Stealth DoubleSpace.)

❑ **QEMM creates and manages the High Memory Area (HMA), the first 64K of extended memory.** This enables certain programs such as DOS or DESQview to store a portion of their code and data in this area.

❑ **QEMM automatically handles programs that can load themselves into upper memory.** Some programs are configured to put themselves in upper memory. Optimize recognizes this, and loads them into High RAM.

√ indicates a new or enhanced feature.

+ indicates a feature not found in MS-DOS 6's memory manager.

+ ❑ **VIDRAM can extend conventional memory up to 96K for running DOS text-based programs.** QEMM's VIDRAM program extends conventional memory into the upper memory address space normally reserved for EGA or VGA graphics. While using VIDRAM, you cannot use EGA or VGA graphics, but you can easily turn VIDRAM on and off, before and after you run your text-based program. VIDRAM also extends conventional memory for DOS text-based programs running in Microsoft Windows. If you have an 8514A video adapter, you can use VIDRAM to extend conventional memory for DOS text-based programs as well as 8514A graphics programs.

❑ **QEMM can create an additional 32K of High RAM from "unused" system ROM.** On certain PCs, if you are not using Stealth ROM, QEMM will find 32K of system ROM that is used only during system bootup, and include those ROM addresses for use as High RAM.

❑ **QEMM creates an additional 32K of High RAM on IBM machines.** IBM computers include an old version of the BASIC programming language in their ROMs. QEMM creates the extra High RAM by mapping memory to the BASIC ROM's address space.

Benefit: QEMM dynamically transforms your PC's memory into whatever kind of memory your programs need.

❑ **QEMM automatically transforms your PC's extended memory (the memory above 1 megabyte) into expanded memory (EMS) when you run programs that use expanded memory.** QEMM is compatible with EMS 3.2, EMS 4, and EEMS and supports EMS 4 real alternate maps.

❑ **QEMM manages extended memory.** QEMM supports all three functions of the XMS v3 specification: HMA (High Memory Area), UMBs (Upper Memory Blocks) and EMBs (Extended Memory Blocks). QEMM provides both VCPI (Quarterdeck/Phar Lap Virtual Control Program Interface) and DPMI (DOS Protected Mode Interface) support. VCPI and DPMI are industry standards which specify how 80386 control programs and protected mode programs coexist. QEMM is fully compatible with DOS-extended programs such as AutoCad 12, Lotus 1-2-3 Release 3, Oracle Professional and Interleaf. (For a technical description of the memory specifications supported by QEMM, see **Appendix C**.)

√ indicates a new or enhanced feature.

+ indicates a feature not found in MS-DOS 6's memory manager.

- **QEMM pools expanded and extended memory and allocates them to programs as needed.** You need not preallocate fixed amounts of expanded and extended memory. QEMM treats all of your PC's memory as a pool of memory—to be used as extended memory or expanded memory at the time your programs need it.

+ - **QEMM provides DPMI (DOS Protected Mode Interface) services and memory for programs that utilize the DPMI 0.9 API** (e.g., Microsoft C/C++ 7.0, Borland C/C++ 3.x, Intel 386/486 C Code Builder Kit). Such programs run in the protected mode of the 386 and 486 processor, allowing them to have direct access to all system memory for both their code and data. QEMM's DPMI Host also provides support for the DPMI 0.9 API under Quarterdeck's DESQview and DESQview/X multitasking environments.

Benefit: QEMM is Microsoft Windows-aware—it detects Windows' presence and optimizes memory for the best Windows performance.

+√ - **QEMM includes a Windows interface**—you can now install and configure QEMM from Microsoft Windows. QEMM also includes a Windows version of Manifest, Quarterdeck's system and memory report tool.

+ - **QEMM can provide 8K-24K more conventional memory for running DOS programs inside Windows' 386 enhanced mode.** QEMM provides space in upper memory for the Windows translation buffers and other parts of Windows (8K-24K) dynamically and automatically, without having to restrict this memory from being used by device drivers and TSRs.

- **By loading TSRs and device drivers into upper memory, QEMM frees up additional memory for running DOS programs under Windows.**

+ - **QEMM's VIDRAM program can extend the amount of conventional memory available to DOS applications running in Windows by as much as 96K.**

Benefit: QEMM is performance- and compatibility-tuned to give you a more smoothly running system.

+ - **QEMM is a full 32-bit protected mode program, giving you maximum memory management performance.**

+ - **QEMM is specially tuned for the Intel Pentium to take advantage of its features for performance and memory savings.**

√ indicates a new or enhanced feature.

+ indicates a feature not found in MS-DOS 6's memory manager.

- **QEMM automatically supports popular disk compressors such as Stacker, SuperStor, DoubleSpace and DriveSpace.** QEMM will also work with XtraDrive.

+√ - **QEMM includes a new version of a Stacker 4 driver which puts Stacker 4's buffer into extended memory, freeing 8-32K in the first megabyte of memory, depending on your configuration.** This new driver utilizes technology developed jointly by Stac Electronics and Quarterdeck Office Systems, and was written by Stac Electronics, the developers of Stacker.

+ - **QEMM supports MS-DOS 6 multiple configurations.** Unlike DOS 6's own memory manager, QEMM lets you select which configuration to optimize on a system with multiple configuration blocks.

- **QEMM supports the Suspend/Resume features of many laptop PCs.** Many battery-powered computers have a Suspend/Resume feature that allows the system to power up in the same state it was in before you turned it off. It resumes the program you were using, as if there were no interruption. If your system supports Suspend/Resume, QEMM will try to detect and enable support of this feature.

- **QEMM can map slow ROM code into faster RAM for better system performance.**

+ - **QEMM supports Adapter Description Libraries for PS/2s.** IBM PS/2s and other Micro Channel architecture (MCA) computers identify each peripheral hardware device with a unique adapter descriptor number. QEMM keeps a record of these descriptions in an Adapter Description Library (ADL) to resolve any memory addressing conflicts with MCA devices.

- **QEMM can free up more memory on certain Compaq computers.** Compaq PCs make a copy of a 16K-32K video ROM in upper memory to speed up video processing. QEMM eliminates this mechanism and compensates for it by shadowing the video ROM with memory mapped over the original ROM location. This creates 16-32K more High RAM and creates a larger contiguous High RAM region. Some Compaqs have a duplicate copy of a 32K system ROM; QEMM eliminates the duplicate copy, creating an additional 32K of High RAM. Some Compaqs use 384K of the extended memory space for top memory, memory used to speed

√ indicates a new or enhanced feature.

+ indicates a feature not found in MS-DOS 6's memory manager.

up ROMs and for Compaq utilities. QEMM performs top memory's functions and adds the 384K of memory to the memory pool so it can be used as expanded or extended memory. QEMM can also shrink down the system ROM size when a Compaq "cut table" is present.

- **QEMM's Optimize program automatically detects adapter ROM and RAM (including RAM mapped into memory after QEMM loads as with many network adapters) to prevent memory conflicts.** Adapters such as network cards, disk controllers and scanner cards generally use specific upper memory addresses. QEMM automatically detects an adapter's address space and prevents it from being used as High RAM. This prevents memory conflicts in an adapter's address space and saves you the trouble of having to tell QEMM which addresses are being used by adapters.

- **QEMM will automatically detect bus-mastering hard drive controllers.** QEMM will detect a bus-mastering hard drive controller and set up a special buffer to prevent memory errors that have been known to occur when using a memory manager along with these devices. QEMM is also fully compatible with the VDS (Virtual DMA Services) specification for resolving compatibility problems between 386 memory managers and bus-mastering devices (for information on the VDS specification, see **Appendix C**).

- **QEMM automatically monitors Direct Memory Access (DMA) into mapped memory to prevent destructive memory writes.**

+ - **QEMM can use shadow memory or top memory to give you more expanded or extended memory.** QEMM can detect Chips & Technologies LEAP, AT386, NEAT, or SCAT ShadowRAM; NEC, OPTI, PEAK or TOPCAT shadow memory, or Compaq-style top memory, and add it to the memory pool, giving you up to 384K more expanded or extended memory.

+ √ - **Optimize will detect PCMCIA adapters and configure memory accordingly.**

+ - **QEMM unleashes the full power of DESQview and DESQview/X.** With QEMM, DESQview and DESQview/X can run programs that write directly to the screen in the background without writing over existing windows, display each program running in a small window on the screen, and protect the system against certain program violations.

√ indicates a new or enhanced feature.

+ indicates a feature not found in MS-DOS 6's memory manager.

+ √ ❑ **The new QuickBoot feature reduces the time it takes to reboot your system.** This feature will save you time when you want or need to do a warm reboot. If you use multiple configurations or operating systems, it will take considerably less time to reboot when you want to change configurations or operating systems. QuickBoot also saves you time if you want to reboot because your system does not seem to be running normally. QuickBoot speeds up the QEMM's Optimize process, which normally reboots your PC two or three times.

Benefit: QEMM provides comprehensive state-of-the-art fine-tuning and analysis tools for power users.

+ √ ❑ **Manifest helps you analyze memory.** QEMM includes Manifest, Quarterdeck's comprehensive, state-of-the-art memory analysis and reporting program. Manifest can help you understand the details of your PC's memory usage. QEMM includes both DOS and Windows version of Manifest. QEMM now includes a Windows version of Manifest, that gives additional information about Windows. The new version of Manifest also gives information about any PCMCIA adapters installed in your system.

+ ❑ **QEMM's LOADHI.COM and QEMM.COM programs can give you information about how High RAM is being used.**

+ ❑ **QEMM has many parameters that let you fine-tune memory management for your particular configuration and needs.**

+ ❑ **Parameter files make it easy to change QEMM configuration.** For advanced users who want to experiment with different QEMM options, parameter files can be a great time saver. Instead of editing the QEMM device line in CONFIG.SYS, you can specify parameters in a file. You can keep separate parameter files for different configurations.

+ ❑ **QEMM's Analysis feature can help you fine-tune memory management for your particular system and programs.** You can run your programs and QEMM will keep track of the memory they use. The Analysis feature will recommend any additional areas of upper memory that can be used as High RAM. The Analysis procedure can also tell you what areas are being accessed by your programs and therefore should not be used as High RAM. This procedure can help you easily troubleshoot any problems that

might occur if a program or piece of hardware needs access to particular upper memory addresses.

√ indicates a new or enhanced feature.

+ indicates a feature not found in MS-DOS 6's memory manager.

+ ❑ **QEMM's Setup program provides an easy way to turn QEMM's various features on and off, troubleshooting tips and an editor for CONFIG.SYS and AUTOEXEC.BAT.**

+ ❑ **Optimize processes batch files called from AUTOEXEC.BAT, loading programs specified in the batch files into High RAM when appropriate.**

+ ❑ **Optimize lets you play "what if" games with the loading order of device drivers and TSRs.** This helps you determine whether changing the loading order will give you more free memory.

+ ❑ **QEMM includes advanced utilities that allow you to manipulate EMS handles.** Advanced users may want to allocate EMS memory temporarily to prevent it from being taken by programs that use or allocate it indiscriminately. The EMS utilities also provide status information about EMS handles.

+ ❑ **QEMM lets you dynamically add DOS FILES and BUFFERS from the DOS prompt for programs that may need more of these resources.**

+ ❑ **QEMM's DPMI Host allows you to specify the amount of virtual memory to be used for programs that use DPMI services.**

+ √ ❑ **The QEMM Programming Interface (QPI) provides programmers with technical information about QEMM's configuration and provides access to many of the capabilities of 80386 control programs.** QPI is documented for the first time in this manual.

About This Manual

QEMM is a collection of programs that help you get the best use of your PC's memory. Each chapter of this guide covers a particular QEMM program or feature. If you are a non-technical PC user, you only need to read the ***QEMM Installation Guide*** which tells you how to install QEMM. If you ever need detailed information about QEMM, the remaining chapters of this manual are your references. At the beginning of each chapter is a brief description of the chapter and why you would need to read it.

If you encounter any problems using QEMM, please see **Appendix A** for troubleshooting information. **Appendix B** contains a list of in-depth technical notes that are copied to your hard disk during QEMM's installation.

IMPORTANT: For late-breaking information that did not make it into this guide, see the READ.ME file in the QEMM directory.

The ***QEMM Installation Guide*** tells you how to install QEMM. The ***QEMM Reference Manual*** (the manual you are reading now) is organized as follows:

Section 1 Using QEMM - how to use QEMM's basic features.

- **Chapter 2** tells you how to use the Setup program to change QEMM's configuration on your system.
- **Chapter 3** tells you how to use the Optimize program.
- **Chapter 4** describes DOS-Up, QEMM's feature that loads parts of DOS into upper memory.
- **Chapter 5** describes Stealth ROM, the feature that creates additional High RAM by hiding ROMs. This chapter also describes Stealth D*Space, the feature that frees up additional memory by hiding DOS 6's DoubleSpace or DriveSpace device driver.
- **Chapter 6** describes VIDRAM, the program that extends conventional memory by as much as 96K for running DOS text-based programs.

Section 2 Technical Reference - a guide to QEMM's technical features, mainly for advanced users who want to fine-tune memory.

- **Chapter 7** describes the optional parameters to QEMM's device driver.
- **Chapter 8** is a technical reference for the LOADHI program that QEMM uses to load items into High RAM.
- **Chapter 9** describes QEMM.COM, a program that can change QEMM's mode, and report on memory usage.

- **Chapter 10** is a technical reference for QEMM's DPMI (DOS Protected Mode Interface) host.
- **Chapter 11** describes QEMM's advanced EMS utility programs.
- **Chapter 12** describes some miscellaneous utility programs included with QEMM.
- **Chapter 13** describes QPI, the QEMM Programming Interface that program developers can use to gain technical information about QEMM's configuration and gives them access to many of the capabilities of 80386 control programs.

Section 3 Appendices

- **Appendix A** is the troubleshooting guide.
- **Appendix B** gives a list of technical notes included in your QEMM directory.
- **Appendix C** gives a technical description of the different memory specifications supported by QEMM.
- **Appendix D** is a glossary of terms related to QEMM and PC memory.

About Manifest

Quarterdeck's Manifest is a standalone program that provides comprehensive memory analysis and reporting functions. Manifest is bundled with QEMM, and QEMM's installation program will automatically install both DOS and Windows versions of Manifest in your QEMM directory. There is no separate Manifest diskette or installation procedure. For information on Manifest, see the ***Manifest User's Guide***.

Using QEMM

2 QEMM Setup: Configuring QEMM

This chapter covers QEMM's Setup program, QSetup, which you can use to configure QEMM on your system. You can also use QSetup to access a wealth of technical information about using QEMM with different hardware and software. Setup does all this and more:

- Enables or disables the features you choose, adding the appropriate lines to your CONFIG.SYS file.
- Adds or removes some of the more commonly-used QEMM parameters and explains what they do and when to use them.
- Provides extensive online help, including detailed explanations of many features and parameters.
- Allows you to manually edit your CONFIG.SYS or AUTOEXEC.BAT files, without leaving QEMM Setup.
- Lets you view hints and tips, the READ.ME file, and a collection of troubleshooting and informational technotes.
- Tells you if you need to run QEMM's Optimize program after changing your configuration—and then runs it for you.

You can run the Setup program from DOS or Microsoft Windows.

Running QEMM Setup from DOS

To run the QEMM Setup program from the DOS prompt:

- **Go to the QEMM directory by typing CD QEMM and pressing Enter ↵.**
- **Type QSETUP and press Enter ↵.**

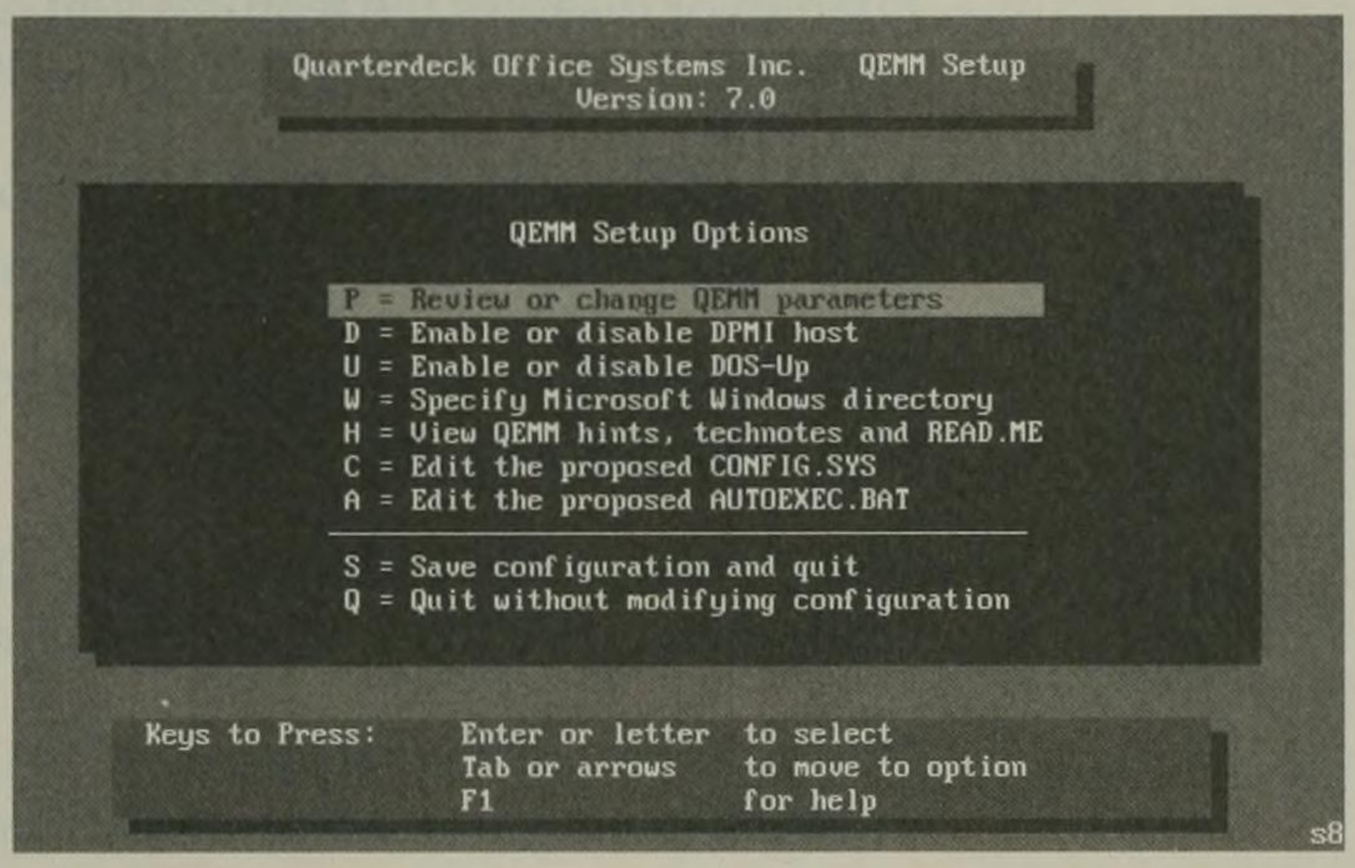
  You will see a welcome screen.
- **Press Enter ↵ to display the QEMM SETUP menu.**
- **If you are using DOS 6 multiple configurations, you will be asked to choose a configuration; select the configuration you want to modify.**

You will see the QEMM Setup menu. To choose an item from the menu:

- **Use the arrow keys or Tab to highlight the item you want, then press Enter ↵. Or, type the letter that appears at the beginning of the item.**

The QEMM Setup menu has the following options:

- **Review or change QEMM's parameters** - adds or changes some of



QEMM Setup - DOS version.

the more frequently-used parameters to the QEMM386.SYS device driver in your CONFIG.SYS file. When you select this option, you will see another menu listing the features you can change. When you select an item, you will see a description of what it does so you can review it before making the change. The menu contains more than one page; use the Page Up and Page Down keys to switch between pages. For detailed information on QEMM's parameters, see **Chapter 7**.

- **Enable or disable DPMI host** - lets you enable or disable the DOS Protected Mode Interface for programs that support DPMI. For information on DPMI, see **Chapter 10**.

- **Enable or disable DOS-Up** - lets you turn on or off QEMM's feature that loads selected parts of DOS into upper memory. You can also partially enable DOS-Up; that is, you can specify which

parts of DOS to load into upper memory. For information on DOS-Up, see **Chapter 4**; also see the online help in QSetup.

- **Enable or disable Stealth D*Space** - turns on or off the feature that hides MS-DOS 6's DoubleSpace or DriveSpace device driver, freeing up 31-49K of RAM. For information on Stealth D*Space, see **Chapter 5**. You will see this selection only if you are using DOS 6's DoubleSpace or DriveSpace feature.

- **Specify Microsoft Windows directory** - lets you specify the path to Microsoft Windows if you are using Windows on your system. When you give the correct path to Windows, then save your configuration, QSetup will add the line SystemROMBreakPoint=false to Windows' SYSTEM.INI file to make Windows compatible with QEMM.

- **View QEMM hints, technotes and READ.ME** - lets you view a wealth of information about QEMM, including descriptions of QEMM's features and tips on using QEMM with various hardware and software. Also included are technotes—technical bulletins containing technical details and troubleshooting information. You can also view the READ.ME file which contains late-breaking information that does not appear in this manual. For an overview of the various technotes, see **Appendix B**.

- **Edit the proposed CONFIG.SYS** - lets you view/edit your CONFIG.SYS file. If you have used QEMM Setup to make changes that affect CONFIG.SYS, you will see those changes. When you are editing the file, you can press F1 to get online Help about the editor's features. Once you have made changes to CONFIG.SYS, you can press F3 to accept the changes or Esc to cancel. The changes will be saved to disk when you select **Save Configuration and Quit** (see below).

- **Edit the proposed AUTOEXEC.BAT** - lets you view/edit your AUTOEXEC.BAT file. If you have used QEMM Setup to make changes that affect AUTOEXEC.BAT, you will see those changes. When you are editing the file, you can press F1 to get online Help about the editor's features. Once you have made changes to AUTOEXEC.BAT, you can press F3 to accept the changes or Esc to cancel. The changes will be saved to disk when you select **Save Configuration and Quit** (see below).

- **Save configuration and Quit** - saves any changes you have made using QSetup and exits.
- **Quit without modifying configuration** - exits the QSetup program, abandoning any changes you have made.

Getting Help

QEMM Setup includes extensive online help. To get online help about any item:

- **Select the item you want to know about, then press F1.**

Running QEMM Setup from Microsoft Windows

To run QEMM Setup from Microsoft Windows:

- **Go to the QEMM group (or the QEMM folder if you are using SideBar).**

  If you do not have a QEMM group (or folder): Create a new program group called QEMM and in that group create a new program item called QEMM Setup using the command line \QEMM\QSETUP.EXE (see your Microsoft Windows documentation for information on creating new items). If you are using SideBar, create a new folder called QEMM, then open a directory view of the QEMM directory and drag the QSETUP.EXE icon to the new folder (see the SideBar manual for information on folders and directory views).

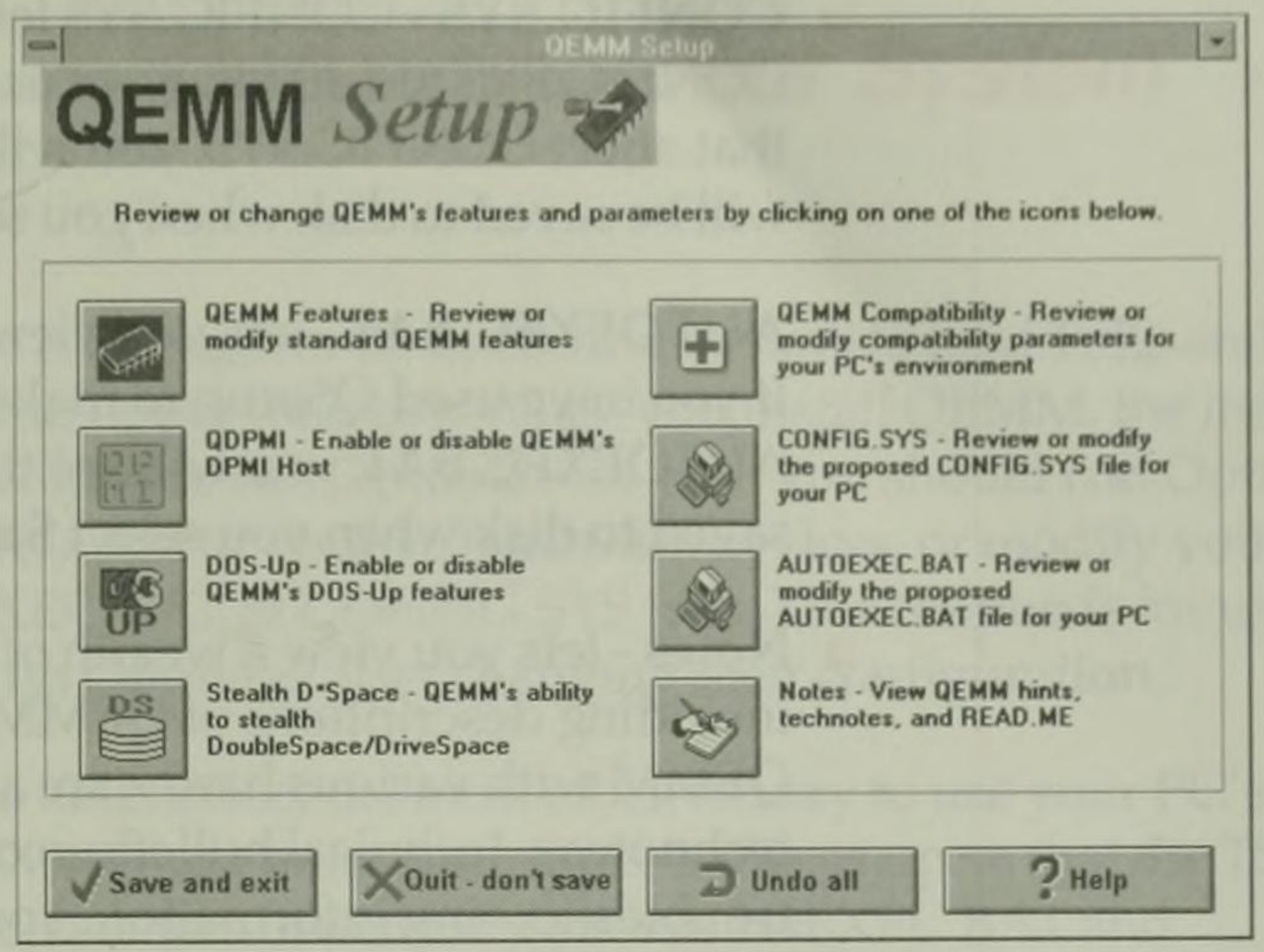
- **Double click on the QEMM Setup icon.**
- **If you are using DOS 6 multiple configurations, you will be asked to choose a configuration; select the configuration you want to modify.**

  You will see the QEMM Setup window.

  Along the top of the screen are buttons you can select to choose the various setup options; when you select one of these buttons, you will be presented with further choices.

  The buttons at the bottom of the screen let you save or abandon your changes and display online help.

  Next, we will give a brief description of the buttons on the QEMM Setup screen. When you select a button you will be presented with further options. For detailed information about any of the options press F1 or click on the Help button for online help.



QEMM Setup - Windows version.

- **QEMM Features** - adds or changes some of the more frequently-used parameters to the QEMM386.SYS device driver in your CONFIG.SYS file. When you select this option, you will see a list of features you can change. To see a description of what a particular option does, move the mouse cursor to the option. For more detailed information select Help. This window also lists QEMM's device line in CONFIG.SYS so you can view it or edit it manually.

- **QDPMI** - This option lets you enable or disable the Quarterdeck DOS Protected Mode Interface for programs that support DPMI. You can also set the size of the DPMI swapfile. For detailed information on QDPMI, see **Chapter 10**.

- **DOS-Up** - DOS-Up lets you turn on or off QEMM's feature that loads selected parts of DOS into upper memory. You can also partially enable DOS-Up; that is, you can specify which parts of DOS to load into upper memory. For information on DOS-Up, see **Chapter 4** or use Help.

- **Stealth D*Space** - turns on or off the feature that hides MS-DOS 6's DoubleSpace or DriveSpace device driver, freeing up 31-49K of RAM. For information on Stealth D*Space, see **Chapter 5** or use Help.

- **QEMM Compatibility** - lets you review, modify or add various QEMM troubleshooting parameters.

- **CONFIG.SYS** - CONFIG.SYS lets you view/edit your CONFIG.SYS file. If you have used QEMM Setup to make changes that affect CONFIG.SYS, you will see those changes. The changes will be saved to disk when you select **Save and exit** (see below).

- **AUTOEXEC.BAT** - lets you view/edit your AUTOEXEC.BAT file. If you have used QSetup to make changes that affect AUTOEXEC.BAT, you will see those changes. The changes will be saved to disk when you select **Save and exit** (see below).

- **Notes** - lets you view a wealth of information about QEMM, including descriptions of QEMM's features and tips on using QEMM with various hardware and software. Also included are technotes—technical bulletins containing technical details and troubleshooting information. You can also view the READ.ME file which contains late-breaking information that does not appear in this manual. For an overview of the various technotes, see **Appendix B**.

- **Save and exit** - saves any changes you have made using QEMM Setup and exits.

- **Quit - don't save** exits the QEMM Setup program, abandoning any changes you have made.

- **Undo all** - restores your CONFIG.SYS and AUTOEXEC.BAT files to the state they were in before you ran QEMM Setup.

- **Help** - displays help for the screen you are currently viewing.

3 Optimizing Your System

Optimize is a program that determines how to load TSRs and device drivers into upper memory. When you install QEMM, the installation program offers to run Optimize for you. You should run Optimize again if you add or remove hardware devices, or modify your AUTOEXEC.BAT or CONFIG.SYS files. This chapter is for users who need to run Optimize after changing their configuration.

Optimize determines the most efficient way to use your PC's memory. Optimize automatically sets things up so that the TSRs and device drivers you load from your AUTOEXEC.BAT and CONFIG.SYS files are loaded into High RAM (the memory that QEMM places between 640K and 1024K) instead of into conventional memory (the area between 0K and 640K). This frees up more conventional memory so you can:

- Run larger programs that would not fit in conventional memory before.
- Add TSRs you have been doing without.
- Speed up programs that were formerly storing their data on disk or on the network because there was not enough available memory to hold the data.
- Increase the conventional memory available to applications running under multitasking control programs (e.g., DESQview, Microsoft Windows).

When to Run Optimize

When you install QEMM, the installation program offers to run Optimize for you. You can also run Optimize later if necessary. You should run Optimize after:

- You add or remove a hardware component.
- You add or remove a device driver or TSR in CONFIG.SYS or AUTOEXEC.BAT.
- You change the loading order of TSRs, or change a TSR command line.

- You change the QEMM386.SYS device line in CONFIG.SYS manually or with QEMM's Setup program.

Optimize will properly handle conditional statements in AUTOEXEC.BAT and DOS 6 multiple configurations.

How to Run Optimize

Before running Optimize, be sure you are at the DOS prompt outside of any multitasking environment such as DESQview or Microsoft Windows. To run Optimize:

- **Type Optimize followed by any optional parameters (explained later in this chapter) and press Enter ↵.**

If you have an LCD, gas plasma or monochrome display, you can run Optimize in monochrome mode by typing **OPTIMIZE /M [...other parameters...]**.

You will be asked to choose between an Express and a Custom Optimize. An Express Optimize optimizes your system automatically. A Custom Optimize lets you monitor the Optimize process and select various options.

During the optimization process your PC must be rebooted two or more times. If it does not reboot when indicated, press the reset button or power the computer off, then on again.

If all of your TSRs, device drivers and selected parts of DOS will not fit in High RAM and you are not already using QEMM's Stealth ROM feature, Optimize will offer to test your system for Stealth compatibility. Stealth ROM creates additional High RAM by mapping memory into ROM addresses (for more information on Stealth ROM, see **Chapter 5**).

During the Optimize process, you will see a screen that tells you how many TSRs and device drivers were loaded high. When Optimize completes, it displays a screen telling you how much more memory you have for running DOS programs. If you would like details on the items that are loaded high, type **LOADHI** and press **Enter ↵** at the DOS prompt.

If You Encounter Problems

If Optimize does not complete successfully or if there are errors when you boot your system after running Optimize, there is a chance that

Optimize loaded a program into an upper memory area used by an adapter (i.e., a network card) or another program. See **Appendix A** for information on running Optimize's automatic exclusion detection (OPTIMIZE /AUTO).

Optimize Options (Custom Optimize Only)

If you choose a Custom Optimize, you will see O for Options at the bottom of the screen during the Setup and Analysis phases. If you type O you will see a list of further options.

If you select Options during the Setup phase, you will see a screen listing the following options:

- **Enter** ↵ causes Optimize to detect upper memory areas that should be excluded from QEMM's management. This option is a troubleshooting procedure that you can use if you ran Optimize and it did not complete successfully. This option is analogous to typing OPTIMIZE /AUTOEXCLUDE. For information, see the AUTOEXCLUDE parameter later in this chapter and the section on Optimize /AUTOEXCLUDE in **Appendix A** for details.

- **F3** tells Optimize to test your system for compatibility with QEMM's Stealth ROM feature.

- **Esc** resumes the normal Optimize process.

If you select Options during Optimize's Analysis phase, you can choose the following:

- **Region Layout** shows which upper memory addresses will be used to load TSRs and device drivers.

- **Modify Data** lets you view and modify the data Optimize uses to load programs high. You can use this screen to load a program low if you know it does not work when loaded high (also see the COMMANDFILE parameter later in this chapter). Do not alter the other fields on this screen unless you know the possible consequences.

- **What-If** lets you see if you can gain memory by changing the loading order of the programs in CONFIG.SYS and AUTOEXEC.BAT. You cannot actually put these changes into effect at this point (and such changes are not always advisable due to dependencies of one program upon another), but you can quickly test different possibilities to see if changing the loading

order of your programs in CONFIG.SYS or AUTOEXEC.BAT is worthwhile.

Forcing Stealth ROM Testing

Optimize offers to test for Stealth ROM compatibility only if it cannot fit all the items in your CONFIG.SYS and AUTOEXEC.BAT files into High RAM. Stealth ROM can typically create 48K-115K of additional High RAM. For a discussion of how Stealth ROM works, see Chapter 5.

If you use DESQview or DESQview/X, you may want to enable Stealth ROM even if Optimize did not recommend Stealth ROM testing. Stealth ROM will usually give DESQview and DESQview/X users more memory for each window. If you have DESQview or DESQview/X, and you are not already using Stealth ROM, we recommend you Optimize using Stealth ROM as follows:

- **Switch to the QEMM directory by typing CD \QEMM and pressing Enter ↵.**
- **Type Optimize /STEALTH and press Enter ↵.**

How Optimize Works

For those who are interested, here is how Optimize works:

- First, Optimize searches your CONFIG.SYS and AUTOEXEC.BAT files and any batch files called by AUTOEXEC.BAT for TSRs (memory resident programs) and device drivers. Optimize makes some changes to CONFIG.SYS and AUTOEXEC.BAT that cause each item to be measured in terms of its initialization size and its size when resident. Your original files are renamed to CONFIG.QDK and AUTOEXEC.QDK.
- Next, Optimize determines how much memory each item needs. Some TSRs and device drivers need more memory to initialize than they do once they are resident in memory, and Optimize notes these differences. Then, Optimize determines the most efficient way to load items into High RAM by testing all possible locations (there can be thousands or millions of possibilities). The object is to leave as much free conventional memory as possible.

  If an item needs a larger amount of memory to initialize than is available in a contiguous chunk, Optimize may use a feature called Squeeze. Squeeze will temporarily map memory over a ROM, adapter RAM, excluded area or the EMS page frame to make a large enough contiguous area, then initialize the item in that area.

Once the item is resident, Squeeze releases any extra memory the item needed to initialize. The Squeeze process is safe—it will never squeeze over memory that the item uses while initializing.

If the items will not all fit into High RAM, Optimize offers to test your PC for compatibility with QEMM's Stealth ROM feature. Stealth ROM temporarily moves ROMs to make more upper memory addresses available for loading items high. If your system is Stealth ROM compatible, Optimize will use it to create more High RAM, then restart the Optimize process.

- Next, Optimize modifies CONFIG.SYS, AUTOEXEC.BAT and any batch files called by AUTOEXEC.BAT to load the TSRs, device drivers and selected parts of DOS into High RAM. To load a device driver high, Optimize adds **DEVICE=C:\QEMM\LOADHI.SYS /R:n SIZE=n** to the beginning of the item's line. To load a TSR high, Optimize adds **C:\QEMM\LOADHI /R:n SIZE=n** to the beginning of the item's line. **LOADHI** is the QEMM program that actually loads items high. The **r:n** parameter (where n is a number) specifies which High RAM region should be used to load the item. The **SIZE=n** parameter tells LOADHI how many bytes of memory the item needs to initialize; if LOADHI cannot find that large of a contiguous area in High RAM, the item will load into conventional memory instead.

During the optimization process your PC must be rebooted two or more times to make the necessary changes to CONFIG.SYS and AUTOEXEC.BAT and load the various items high.

Once Optimize is finished, the items will automatically load into High RAM whenever you boot your PC.

Undoing an Optimize

You can restore your CONFIG.SYS and AUTOEXEC.BAT files to the state they were in before you most recently ran Optimize. To undo the most recent Optimize:

- **Type UNOPT and press Enter ↵.**

  CONFIG.SYS and AUTOEXEC.BAT will be restored to the state they were in before you last ran Optimize. If your AUTOEXEC.BAT file calls another batch file, that batch file will be restored to its pre-optimized state.

If you have run Optimize more than once, you can restore up to ten previous configurations (i.e., CONFIG.SYS and AUTOEXEC.BAT), including the configuration that existed before you installed QEMM. For information, see the /RESTORE parameter in the section, "Optimize Parameters," later in this chapter.

Optimizing Embedded Batch Files

Optimize can detect batch files embedded in AUTOEXEC.BAT. To take advantage of this capability, be sure that any line in the AUTOEXEC.BAT that starts another batch file is preceded by the CALL command. You can nest batch files as many levels as you want (e.g., AUTOEXEC.BAT can call a batch file that calls a batch file, etc.).

For example, if you have a batch file in AUTOEXEC.BAT that logs you into the network such as C:\NETWORK\STARTNET.BAT, you would change this line to CALL C:\NETWORK\STARTNET.BAT.

The CALL command is supported by the COMMAND.COM shell for DOS versions 3.30 and above, and for most popular replacement command interpreters.

Optimize and MS-DOS 6 Multiple Configurations

MS-DOS 6 allows you to specify several alternative configurations in your CONFIG.SYS file. When you use this feature, you will see a startup menu listing the different configurations each time you boot your PC. For information on multiple configurations, see the DOS 6 manual.

When you run Optimize, it will detect any multiple configurations you have set up and will display a menu listing the same selections as DOS's startup menu. Optimize will process the configuration you select. Optimize will then execute normally, booting the system with the configuration that you have chosen. To optimize another configuration, you will need to run Optimize again and select the configuration you want.

Optimize will modify your CONFIG.SYS and AUTOEXEC.BAT files, adding the appropriate LOADHI commands to load device drivers and TSRs high. When processing multiple configurations, Optimize adds the LOADHI parameter /RF (/RESPONSEFILE) instead of the parameter /R:n (/REGION) (for information on these parameters, see "The LOADHI Parameters" in **Chapter 8**).

For more information on QEMM's handling of multiple configurations, see the MSDOS6.TEC technote included with QEMM (**Appendix B** contains information about accessing the technotes).

Excluding TSRs or Device Drivers from Optimize

If you have device drivers or TSRs in CONFIG.SYS or AUTOEXEC.BAT that you would like to exclude from the Optimize process (i.e., you want them to load into conventional memory), you can do so as follows:

- **Create an ASCII text file called OPTIMIZE.NOT in the QEMM directory.**

Earlier versions of QEMM called this file OPTIMIZE.EXC instead of OPTIMIZE.NOT. The name OPTIMIZE.EXC is still supported to be compatible with earlier versions.

- **List each program you want to exclude on a separate line in OPTIMIZE.NOT. When you specify a command or program in this file, use only the first part of the filename; do not specify the drive, path, extension or any parameters (e.g., for the file C:\UTILITY\THISPROG.EXE, you would specify only THISPROG).**
- **Save the file.**
- **Run Optimize.**

  Optimize will exclude the programs you listed from being loaded high. Also see the COMMANDFILE parameter in the next section.

UMB-using Programs and Optimize

Some TSRs and device drivers can load themselves into upper memory. We call these programs *UMB-using programs*. Usually, such a program has an optional parameter that causes it to load into a UMB (upper memory block). If you have a UMB-using program and you want it to load into upper memory, see the program's manual for information on how to load it into a UMB. Optimize will let the program load into a UMB and will take this into account during its memory calculations.

If you have a UMB-using program, but you want it to load into conventional memory, see the program's manual for any parameters you need to include (or leave off) to force the program not to use UMBs. Then, add the program name to the file OPTIMIZE.NOT (see

the preceding section). Do not add a program to the OPTIMIZE.NOT file if the program is going to load itself into upper memory; doing so will cause Optimize problems.

If you use a command processor that loads parts of itself into UMBs (e.g., 4DOS and NDOS, when they are configured to do so), Optimize will make the proper use of upper memory only if you use QEMM's DOS-Up feature to load the COMMAND.COM into upper memory (QEMM's DOS-Up feature does this by default). If you experience any problems with DOS-Up loading a UMB-using command processor into upper memory, see *page 180* in **Appendix A** for information.

Optimize Parameters

Listed below are the parameters you can use if you are running Optimize from the DOS prompt. Most of the parameters will be of interest only to advanced users. Some parameters have an abbreviation you can use instead of the full name; abbreviations are listed in parentheses below. Options described as "internal" are for Optimize's use—there is no reason for you to specify them.

You can display the list of parameters by typing **OPTIMIZE /?** or **OPTIMIZE /HELP**.

- **/ALLOWENV (/AE)**

  Normally, Optimize ignores any AUTOEXEC.BAT command that starts an operating environment, because you would have to intervene in the Optimize process and exit the environment so that Optimize can continue. Once Optimize is finished, a command that starts an operating environment will be executed. (By the way, if Optimize tests your system for Stealth ROM compatibility, a command that starts an operating environment will be executed to test the operating environment's compatibility with Stealth ROM. When the operating environment starts, just exit the operating environment and Optimize will continue.)

  Use the /ALLOWENV parameter if you want your operating environment to start each time Optimize reboots your system. You may want to use this parameter if Optimize mistakes one of your TSRs for an operating environment and loads it low. The /ALLOWENV parameter will allow processing of the following commands: DV, DVX, XDV, DOSSHELL, NC, PCSHELL, SHELL, SHELLB, SHELLC, WIN, WIN386, WIN86, WINDOWS, and COMMAND (to start a secondary command processor). If you use

/ALLOWENV, be sure to exit your operating environment whenever Optimize reboots so Optimize can continue.

- **/AUTOEXCLUDE (/AUTO)**

  tells Optimize to load all CONFIG.SYS and AUTOEXEC.BAT items low temporarily, then look for areas of upper memory that are accessed during the boot process. Optimize will then offer to exclude these areas from QEMM's management. The purpose of this parameter is to find upper memory areas that are accessed during the boot process so you can prevent QEMM from loading other items there. For example, a network adapter may use a particular area of upper memory, and if QEMM places High RAM there, the network will not work.

  When you use OPTIMIZE /AUTOEXCLUDE, you will be asked to power your PC off, then on. Then, Optimize will present you with a list of the accessed upper memory areas and tell you which programs use them. You can then decide whether to have Optimize exclude any of those areas from QEMM's management. We suggest you use AUTOEXCLUDE if you previously ran Optimize and it did not complete successfully, or if your system did not initialize properly after running Optimize. For more information on using OPTIMIZE /AUTOEXCLUDE, see **Appendix A**.

- **/BOOT:drive (/B:drive)**

  tells Optimize where to find the CONFIG.SYS and AUTOEXEC.BAT files. The default is the root directory of your hard disk, usually the C drive. You use the BOOT parameter to tell Optimize where to look for these files (replace **drive** with the drive letter) when you boot from a floppy or a drive other than C.

- **/COMMANDFILE=filename (/CMD=filename)**

  specifies a file that lists commands or programs which Optimize should not process. Use this parameter when you do not want Optimize to attempt to load certain programs high (e.g., programs that you know will not run properly when loaded high or commands specific to an alternate command processor such as 4DOS or NDOS).

  The file of commands and programs should be an ASCII file with one command or program per line in the file. When you specify a command or program in this file, use only the first part of the filename; do not specify the drive, path, extension or any parameters (i.e., for the file C:\UTILITY\THISPROG.EXE, specify only

THISPROG). If you insert the # symbol at the beginning of a line, that line will be ignored.

If you are using a command processor other than COMMAND.COM (e.g., 4DOS, NDOS) and your AUTOEXEC.BAT file contains commands specific to that command processor, Optimize may misinterpret the commands. QEMM includes a file called 4DOS.CMD which contains a list of 4DOS- and NDOS-specific commands. If you are using 4DOS or NDOS, you can add **/CMD=C:\QEMM\4DOS.CMD** to the Optimize command line—this parameter tells Optimize to ignore the commands listed in this file.

There is an easier method for excluding commands or programs from being processed by Optimize. Just create an ASCII file called OPTIMIZE.NOT in the QEMM directory, and in it, include the commands and programs you want Optimize to ignore. Use the same format as described above. The advantage of using OPTIMIZE.NOT is that you do not have to use the /CMD parameter; whenever you run Optimize, it will check this file. If you have 4DOS or NDOS, you may want to copy 4DOS.CMD to OPTIMIZE.NOT so the 4DOS or NDOS commands will be automatically ignored whenever you run Optimize. If you use the OPTIMIZE.NOT file and use the CMD parameter to specify another file, the programs in both files will be ignored by Optimize.

Placing a program in a command file will force a program to load low. So, if a line already has the LOADHI command, the command file will force Optimize to strip it off.

- **/CONFIG:block_name (/C:block_name)**
  tells Optimize to process the CONFIG.SYS path specified by block_name. CONFIG.SYS paths are an element of the DOS 6 multiple configuration feature; each path begins with a block name enclosed in brackets. You can use the /CONFIG parameter if you are using MS-DOS 6 multiple configurations and you want Optimize to process a specific CONFIG.SYS path automatically. If you do not specify this parameter, Optimize will present you with a menu listing all of the paths, and you can choose the one you want.

  When you use the parameter /CONFIG:block_name, be sure to replace block_name with the label of the path that you want Optimize to process. The label is the name that is enclosed in square brackets at the beginning of the block in CONFIG.SYS.

- **/DIRECTORY (/DIR)**
  specifies the directory Optimize uses to store its information.

- **/DOSUP:n (/D:n)**
  tells Optimize to break DOS into n blocks for purposes of loading it into High RAM with QEMM's DOS-Up feature. DOS-Up treats DOS as several components (usually between five and ten, depending on the version and make of your DOS) that can be placed in different areas of memory. Optimize could load each of these components into a separate region of High RAM, but this would cause Optimize to take an unacceptably long time on many systems, because of the great number of combinations of programs and High RAM regions that Optimize would have to calculate. Therefore, by default, Optimize groups these DOS pieces into n units (n is chosen on the fly to keep the number of combinations lower than 100,000,000, if possible) and treats each unit as a single entity for purposes of loading DOS into High RAM.

  The /DOSUP:n switch lets you choose the number of units into which Optimize will group the parts of DOS. If you want to see if Optimize can free more conventional memory, and you do not care how long Optimize takes to complete, you can use the parameter /DOSUP:0, which tells Optimize to load each DOS-Up block as a separate entity. (The number 0 serves a special function for this parameter, and creates the same effect as if you used a very high value of n.) On the other hand, if you want to reduce the time that Optimize takes to perform its calculations, and are willing to risk getting less conventional memory from Optimize, you can use the parameter /DOSUP:1. With the exception of the number 0, lower numbers cause Optimize to take less time and to be somewhat less efficient; higher numbers (and 0) cause Optimize to be more efficient and to take more time.

- **/EMM:filename**
  specifies the name of the memory manager you are using in case Optimize does not recognize the presence of the memory manager's driver. You should use this parameter if you have for some reason renamed the QEMM386.SYS driver.

- **/FILE (/F)**
  is a command used internally by Optimize.

- **/FINISHED (/DONE)**
  is a command used internally by Optimize.

- **/HELP**
  tells Optimize to list a brief description of each of its parameters.

- **/LARGE (/LA)**
  tells Optimize to display its screens using 43- or 50-line mode if your video adapter supports this feature.

- **/LOADHIONLY (/L)**
  tells Optimize to analyze or change only those lines that already contain a LOADHI command.

- **/LOADLOW (/LOW)**
  causes Optimize to remove all LOADHI statements and /REGION (/R) parameters. **/LOADLOW** forces all programs to load into conventional memory.

- **/MONO (/M)**
  causes Optimize to run in monochrome mode. This parameter is useful for running Optimize on systems with an LCD, monochrome or gas plasma display.

- **/NOARAM (/NA)**
  tells Optimize not to detect adapter RAM automatically. This is a troubleshooting parameter. You may use /NOARAM if Optimize fails and you suspect there may be a problem with adapter RAM being incorrectly detected.

- **/NODOSUP (/ND)**
  tells Optimize not to enable QEMM's DOS-Up feature on your system. Normally, Optimize automatically enables DOS-Up. /NODOSUP is a troubleshooting parameter you can use if you think that loading DOS into High RAM is causing problems on your system.

- **/NOFLUSH (/NOFL)**
  tells Optimize not to flush delayed-write disk caches before rebooting your system. By default, Optimize tries to make delayed-write disk caches write all data to the hard disk before each reboot, so that

Optimize's changes to various files on the disk will not be lost. /NOFLUSH is a troubleshooting parameter you can use if you think that Optimize's attempts to flush disk caches are causing problems on your system. If you use the /NOFLUSH option and you have a disk cache, you should turn off the delayed-write feature of the cache before running Optimize. You may re-enable the delayed-write feature after Optimize completes.

- **/NOSHELL (/NOSH)**

  tells Optimize not to try to load your command processor into High RAM. As part of QEMM's DOS-Up feature, Optimize normally tries to load the command processor high. /NOSHELL is a troubleshooting parameter you can use if you think that loading the command processor into High RAM is causing problems on your system. If you have disabled the DOS-Up feature (either with the /NODOSUP switch or through QEMM Setup), you do not need this parameter.

  Do not use the /NOSHELL parameter if your command processor loads parts of itself into upper memory blocks (UMBs); 4DOS and NDOS are examples of command processors that can use UMBs, depending on how they are configured. If your command processor uses UMBs and you want to prevent Optimize from loading it high, start Optimize without the /NOSHELL parameter. Then, select a Custom Optimize, let it proceed to the Analysis Complete screen and select Options. Then select the Modify Data option and place an N next to the name of your command processor in the field **Try to Load High?**. You can then let Optimize proceed to its conclusion. This method lets Optimize calculate the amount of available memory properly with a UMB-using command processor present.

- **/NOSQF (/NF)**

  tells Optimize you do not want LOADHI to Squeeze using the page frame during program initialization.

- **/NOSQT (/NT)**

  tells Optimize that you do not want LOADHI to Squeeze using temporarily mapped memory during program initialization.

- **/NOSTEALTH (/NOST)**

  tells Optimize not to test your PC for compatibility with Stealth ROM. Without this parameter, Optimize will ask if you want to test

for Stealth ROM compatibility if all items will not fit into upper memory. When used with the /QUICK parameter, the /NOSTEALTH parameter speeds up optimization by eliminating the prompt for Stealth ROM testing.

- **/NOSYNC (/N)**

  tells Optimize your display does not require synchronization. You should use this parameter if you have a CGA display that does not require synchronization (i.e., it does not have "snow effects").

- **/PATH (/P)**

  tells Optimize to add the QEMM directory to the PATH environment variable in your AUTOEXEC.BAT file. This is helpful when the LOADHI lines in your AUTOEXEC.BAT file become too long (i.e., more than 128 characters). This can occur when Optimize adds the LOADHI command to a long line in AUTOEXEC.BAT.

- **/QUICK (/Q)**

  tells Optimize to use the default options.

- **/RESTORE (/R)**

  tells Optimize to post a list of the ten most recent configurations (each configuration includes the CONFIG.SYS, AUTOEXEC.BAT, and called batch files) that have been modified by QEMM programs. You can then select which configuration you would like Optimize to restore. The oldest configuration saved (normally the configuration that existed before you installed this version of QEMM) will always remain on the list of past configurations, even if QEMM programs have made more than ten changes to your configuration.

  If the Optimize program is not located in the same directory as the rest of the QEMM files, you must use the DIRECTORY parameter, as well as the RESTORE parameter, to tell Optimize where to find its information about the configuration history.

- **/RESPONSE (/RF)**

  tells Optimize to prepare a "response file" for subsequent use by the LOADHI programs. /RESPONSE may be followed by a colon and a filename. The default filename is LOADHI.RF. This response file is essentially a database defining your system's use of High RAM. It will contain the detailed specifications for each device driver and TSR to be loaded into High RAM.

This option will be valuable if your PC is connected to a network and your AUTOEXEC.BAT file calls public batch files located on the network. Such public batch files are commonly called by multiple network users. LOADHI information generally must not be specified inside the public batch file, as each PC will likely require a different set of LOADHI parameters. By giving Optimize the /RESPONSE parameter, network PC users can create separate, private response files that contain the LOADHI information specific to each PC.

You may simply wish to use the /RESPONSE parameter to remove the clutter of details from CONFIG.SYS and AUTOEXEC.BAT, especially if the LOADHI command lines become too long to be processed correctly.

- **/SEGMENT (/SEG)**
  is a command used internally by Optimize.

- **/STEALTH (/ST)**
  causes Optimize to test your system for compatibility with Stealth ROM. Optimize will first try the Stealth ROM mapping method and, if it is appropriate for your system, Optimize will implement that method. If the mapping method is not right for your system, Optimize will try the frame method.

- **/STPASSDONE**
  is a command used internally by Optimize.

- **WINCOLORS (/WINC)**
  is a command used internally by Optimize.

- **/?**
  tells Optimize to list its command line parameters.

NOTES:

4 DOS-Up:
Loading Parts of DOS Into Upper Memory

DOS-Up is a QEMM feature that frees up conventional memory by loading selected parts of DOS into upper memory. QEMM's installation program enables DOS-Up on your system by default. Read this chapter if you want technical details on DOS-Up or you need to enable or disable it.

QEMM's DOS-Up feature loads selected parts of DOS into High RAM. Without DOS-Up, these parts of DOS would be loaded in conventional memory (if you have DOS 5 or 6, some but not all of these parts of DOS can be loaded into the HMA). Depending on how your system is configured, DOS-Up can free up 7K to 70K of conventional memory, leaving more room to run your programs.

DOS-Up is normally installed and enabled when you install QEMM. This chapter describes DOS-Up and QEMM's DOS Resource programs which provide an alternative method for loading parts of DOS high. We also tell you how to enable DOS-Up if you did not do so at installation time.

For information on using DOS-Up with DR DOS or Novell DOS, refer to the DR-DOS 6 or Novell DOS technote provided with QEMM. See **Appendix B** *for information on accessing these technotes.*

DOS-Up can load the following components of DOS into High RAM:

- The DOS kernel. This is the "core" of DOS's code and is contained in the system file, MSDOS.SYS (IBMDOS.COM if you have IBM DOS).
- DOS data (contained in MSDOS.SYS or IBMDOS.COM)
- DOS FILES, BUFFERS, STACKS, FCBS and LASTDRIVE. The sizes of these resources are normally specified in CONFIG.SYS, and have default values, if not.
- The DOS command processor, COMMAND.COM.

The amount of conventional memory freed up by DOS-Up depends on your system configuration and what version of DOS you are using. For example, each DOS BUFFER normally takes up approximately half a K of memory. If you have a BUFFERS=30 statement in your CONFIG.SYS, DOS-Up can save 15K of conventional memory by loading the BUFFERS into High RAM.

DOS versions 5 and 6 can load portions of DOS into the HMA (the first 64K of extended memory). To enable this feature, you need the line DOS=HIGH in your CONFIG.SYS file (QEMM's installation may have already put it there for you). Even if you use this feature of DOS, we still recommend using DOS-Up—it will load additional portions of DOS into High RAM.

In addition to the parts of DOS that DOS=HIGH loads into the HMA, DOS-Up can load the following parts of DOS into High RAM: 5K of DOS data, an additional 2.5K of the command processor, FILES, STACKS, FCBS and LASTDRIVE.

If you are using DOS version 3 or 4, DOS-Up will be a great benefit because these versions of DOS load all parts of DOS into conventional memory. DOS-Up can free up to 70K of conventional memory on DOS 3 and 4 systems.

If You Have DESQview or DESQview/X

In general, DESQview and DESQview/X make more efficient use of the HMA than DOS—they are designed to load over 63K of code into this 64K area. That is why we recommend letting DESQview or DESQview/X use that area. DESQview or DESQview/X will automatically use the HMA if no other program is already using it. To have DESQview or DESQview/X use the HMA, just omit the DOS=HIGH statement from CONFIG.SYS. Then, to load selected parts of DOS high, enable DOS-Up (see below) and run Optimize.

Enabling or Disabling DOS-Up

You can enable or disable DOS-Up form the QEMM Setup program. We will explain how to do this in the DOS version of QEMM Setup, though you could use the Windows version if you like (for information on using the Windows version, see **Chapter 2**). To enable or disable DOS-Up:

- **Start QEMM's Setup program by typing QSETUP and pressing Enter ↵. Then, press Enter ↵ to go to the Setup menu.**

- **Select Enable or disable DOS-Up, then select Yes to enable it, No to disable it, or Partial to select which parts of DOS you want to load high.**

  If you choose Partial, you will see a menu where you can select which parts of DOS you want to be loaded high.

If you want more information on DOS-Up, including a description of the selected parts of DOS you can choose to load high, press F1 for Help. Press PageDown to scroll through the Help screens.

Once you have made your DOS-Up selection, you will be back at the QEMM Setup menu.

- **Select Save Configuration and quit.**

  Certain changes to DOS-Up will not be implemented unless you run Optimize. If you see the message, "You have made changes that require the Optimize program to be run":

- **Press Enter ↵ to start Optimize.**

DOS-Up Drivers

DOS-Up consists of two device drivers called DOSDATA.SYS and DOS-UP.SYS which are loaded in your CONFIG.SYS file. For DOS-Up to be fully effective, both drivers must be loaded.

The DOSDATA.SYS device driver prepares your system for the DOS Data area to be loaded into High RAM. When DOS-Up is enabled, the line

DEVICE=C:\QEMM\DOSDATA.SYS

should appear as the first device driver line in your CONFIG.SYS file.

The DOS-UP.SYS device driver is responsible for loading DOS data, the DOS kernel, BUFFERS, FILES, FCBS, STACKS, and LASTDRIVE high. QEMM's Optimize program will determine where to load these items in High RAM. The line

DEVICE=C:\QEMM\DOS-UP.SYS @C:\QEMM\DOS-UP.DAT

should appear immediately after the QEMM386.SYS device driver line in your CONFIG.SYS file. The file DOS-UP.DAT contains

instructions that tell DOS-Up where in High RAM to load the various parts of DOS. You should not modify that file.

To load DOS's command processor (COMMAND.COM) high, QEMM's Setup adds the LOADHI command syntax to the SHELL statement in your CONFIG.SYS file. If you do not have a SHELL statement, DOS-Up will add one. For example, if your original SHELL line was:

SHELL=C:\DOS\COMMAND.COM C:\DOS\ /P

DOS-Up will modify it to read:

SHELL=C:\QEMM\LOADHI.COM /R:n
C:\DOS\COMMAND.COM C:\DOS\ /P

where x is the number of the High RAM region into which COMMAND.COM is loaded. (The entire command above will appear on a single line in CONFIG.SYS.)

DOSDATA Parameters

DOS-Up's DOSDATA driver includes a few parameters that may be needed in special situations. You may add any of the following parameters to the DOSDATA.SYS line if necessary:

- **ALTVIDEO- (V-)**
  a required parameter if you are using the password feature of Stacker version 3.1 or later. These versions of Stacker post a copyright notice and optionally ask the user for a password before the CONFIG.SYS file begins to load.

  DOSDATA detects the presence of Stacker 3.1 or later, and automatically suppresses all BIOS TTY video output while it is reloading DOS into memory. DOSDATA does this so that the Stacker copyright notice will not appear on screen twice. However, if you are using the Stacker password feature, and DOSDATA suppresses Stacker's video output, you will not see the password prompt the second time Stacker asks for the password, and your system will appear to be hung (even though entering the password will allow you to proceed). The ALTVIDEO- parameter allows Stacker's copyright notice and password prompt to display both times.

- **BOOTFILE=pathname (F=pathname)**
  tells DOSDATA where to find a file that contains a boot record. By default, DOSDATA looks for the file BOOTSECT.DOS (for compatibility with Microsoft Windows NT), then loads the boot record from the boot disk if BOOTSECT.DOS is not present. If a boot manager program has created a file to replace the boot record, the BOOTFILE parameter can prevent symptoms such as the boot manager's menu appearing twice instead of once, or the need to reboot twice before the system's DOS partition is accessed. For example, the PC Magazine program MBOOT requires the parameter BOOTFILE=C:\LASTBOOT.BIN. If you want to tell DOSDATA not to look for a file containing a boot sector, use the parameter BOOTFILE=NUL.

- **FORCEBIOS+ (X+)**
  tells DOSDATA to allow the BIOS to load the boot drive's boot record, instead of loading the boot record itself. By default, DOSDATA speeds up the boot process by using its own method of loading the boot record. DOSDATA knows how to load the boot record on hard disks that use Stacker, DoubleSpace, DriveSpace, SuperStor, and XtraDrive, but you may need to use the FORCEBIOS+ parameter on systems with some unusual form of disk compression. You can also try FORCEBIOS+ for DOSDATA problems that may relate to an unusual hard disk or disk controller, or when the symptom of a problem is a "Non-system disk or disk error" message.

- **INITBASE=xxxx (B=xxxx)**
  tells DOSDATA where it should place itself in conventional memory while it is reloading DOS. xxxx represents a number of paragraphs (units of 16 bytes); DOSDATA will use this number to adjust its calculation of what addresses it should occupy. During the boot process, DOSDATA places itself somewhere near the top of conventional memory. The area above DOSDATA must be big enough to initialize DOS, and the area below must be big enough to load device drivers. DOSDATA calculates an optimal location for itself, so you should not normally need to use the INITBASE parameter. However, depending on the configuration of your system, there is a small possibility that DOS may need more room at the top of conventional memory than DOSDATA has allowed, in which case the system will fail, usually while the DOSDATA logo is on the screen. The bigger the value of INITBASE, the lower in conventional memory DOSDATA will place itself, and the more room DOS will

have to initialize. Values of xxxx that are commonly tried are 1001 and 2001.

- **IODATA- (I-)**
  tells DOSDATA not to prepare selected portions of the DOS IO.SYS (or IBMBIO.COM on IBM DOS systems) file to be moved to upper memory. By default, DOSDATA moves about 1K of IO.SYS data structures, containing disk partitioning information and other disk I/O characteristics. When you use the IODATA- parameter, DOSDATA does not move these data structures, at a cost of 1K of conventional memory. IODATA- is rarely needed, but can be tried if you are having problems accessing floppy or hard disks.

QEMM's DOS Resource Programs

Earlier versions of QEMM did not feature DOS-Up, but did include programs that could load certain DOS resources high. To be backward-compatible with earlier versions of QEMM, we have included these DOS resource programs with QEMM version 7.

DOS-Up provides a better and more comprehensive method for loading parts of DOS high, so we recommend using DOS-Up whenever possible. If you are using DOS-Up, you will not need to use the programs described in the remainder of this chapter.

QEMM includes four programs you can use to load additional FILES (file handles), BUFFERS, FCBS (file control blocks), and logical disk drives on the fly, without modifying CONFIG.SYS or rebooting your PC. QEMM's programs to augment these resources are BUFFERS.COM, FILES.COM, FCBS.COM and LASTDRIV.COM.

These programs all work the same way. If you run the program without parameters, you get a report on the resource's current allocation. Each program can take a numeric parameter to increase the resource. You can also use them along with the LOADHI command to load the resource high, thus freeing up more conventional memory to run DOS programs. The additional resources will stay in effect until you power off your PC or reboot.

The programs use the same parameter syntax. A number, or a number preceded by an equals sign (**nn** or **=nn**), specifies the total number of resources needed. A number preceded by a plus sign (**+nn**), adds that many more resources. All the programs add

resources; you cannot use them to reduce the number of resources allocated. You run these programs from the DOS prompt.

BUFFERS.COM

The DOS BUFFERS resource can improve disk I/O response times. Most users have a BUFFERS statement in their CONFIG.SYS file (e.g., BUFFERS=30), but if there is none, DOS allocates a certain number of buffers (usually 15) by default. Each buffer normally uses 528 bytes of memory, so 30 BUFFERS take up about 15K of conventional memory.

By adding BUFFERS, you can improve some programs' response times, but additional BUFFERS also reduce the memory available to programs. Not all programs benefit from additional BUFFERS. By having BUFFERS in High RAM you can have the benefits of buffering without giving up conventional memory.

To display the number of buffers now allocated:

- **Type BUFFERS and press Enter ↵.**

  A message will display telling you how many BUFFERS are allocated.

  To increase the number of BUFFERS so that a total of **nn** BUFFERS are allocated:

- **Type BUFFERS=nn and press Enter ↵. Substitute the number of BUFFERS you want for nn.**

  To load an additional nn BUFFERS into High RAM:

- **Type LOADHI BUFFERS +nn and press Enter ↵. Substitute the number of additional BUFFERS you want for nn.**

FILES.COM

DOS's FILES (file handles) resource sets the number of files that can be open at any one time and requires about 53 bytes for each file handle. You may have a FILES= statement in your CONFIG.SYS file specifying a certain number of file handles (DOS 6's default is 8).

To see how many files are currently allocated:

- **Type FILES and press Enter ↵.**

  To increase the number of FILES so that a total of **nn** FILES are allocated:

- **Type FILES=nn and press Enter ↵. Substitute the number of FILES you want for nn.**

To load an additional nn FILES into High RAM:

- **Type LOADHI FILES +nn and press Enter ↵. Substitute the number of additional FILES you want for nn.**

FCBS.COM

DOS uses the FCBS resource to keep track of File Control Blocks. Some older programs use FCBS instead of file handles to manage open files.

In DOS versions 5 and 6, the FCBS statement includes a number that tells DOS to allocate memory for that many FCBS.

In DOS versions 3 and 4, the FCBS statement has two numbers: the first tells DOS to allocate memory for that many FCBS; the second specifies how many FCBS to protect when DOS needs to close an open FCB. Each FCBS resource uses 53 bytes.

To see how many FCBS are now allocated:

- **Type FCBS and press Enter ↵.**

To increase the number of FCBS so that a total of **n** FCBS are allocated with x additional FCBS protected:

- **Type FCBS=n,x and press Enter ↵. Substitute the number of FCBS you want for n and the number of protected FCBS for x. If you are using DOS 5 or 6, omit the second number (x).**

To load an additional **n** FCBS with x protected FCBS into High RAM:

- **Type LOADHI FCBS +n,x and press Enter ↵. Substitute the number of additional FCBS you want for n and the number of protected FCBS for x. If you are using DOS 5 or 6, omit the second number (x).**

FCBS use contiguous memory. When you add more FCBS, a new block of memory for all FCBS must be allocated in High RAM and the memory used for the original FCBS is not recovered.

LASTDRIV.COM

DOS uses the LASTDRIVE resource to support both physical and logical disk drives. Logical drives are useful if you use the DOS SUBST program or certain networks. DOS maintains a table of drives and each table entry requires about 100 bytes.

To see how many drives are now allocated:

- **Type LASTDRIV and press Enter ↵.**

If you want to allocate additional logical drives for networks or the SUBST command, we suggest changing the LASTDRIVE= statement in CONFIG.SYS to specify only your actual disk volumes (this conserves memory). If your hard disk is partitioned, specify the last partition (e.g., LASTDRIVE=D). Then, load logical drives into High RAM by adding a statement to your AUTOEXEC.BAT file. For example, to add three logical drives, add the following statement to AUTOEXEC.BAT:

```
C:\QEMM\LOADHI  C:\QEMM\LASTDRIV +3
```

You can add additional logical drives using a plus sign and a number, as above, or you can set it using a drive letter (e.g., C:\QEMM\LOADHI C:\QEMM\LASTDRIV=G). If you are using a number, you must include the plus sign.

The DOS drive table uses contiguous memory. When you add more drives, a new block of memory for the entire table must be allocated in High RAM and the conventional memory used for the original drive table (approximately 100 bytes per drive) is not recovered.

NOTES:

5 Stealth ROM and Stealth D*Space: Maximizing High RAM

Stealth ROM is an exclusive QEMM feature that can typically create an additional 48K to 115K of High RAM on almost any PC. Stealth ROM hides your PC's ROMs and makes their memory addresses available for High RAM or expanded memory mapping. Stealth D*Space hides MS-DOS 6's DoubleSpace or DriveSpace device driver, freeing up 31K-49K of memory. Depending on your configuration, QEMM's installation or Optimize may have enabled these features on your system. Read this chapter if you want to know how Stealth ROM or Stealth D*Space works, or you want to enable either of these features.

Enabling Stealth ROM

Depending on your configuration and the installation options you chose, Stealth ROM may have been enabled on your system when you installed QEMM. When you run QEMM's Optimize program, Optimize will try to load your TSRs, device drivers and selected parts of DOS into High RAM. If all of them will not fit, Optimize will test your system for compatibility with Stealth ROM and will determine which Stealth ROM method is best for your system.

Very few systems are incompatible with Stealth ROM—Optimize will not enable Stealth ROM if it determines that your system is not Stealth-compatible. Also, Stealth ROM depends on the existence of the EMS page frame—if you want to use Stealth ROM, you must do nothing that would eliminate or reduce the size of the page frame (i.e., do not use the parameters EMS:N; FRAMELENGTH=0, 1, 2 or 3; or FRAME=NONE on the QEMM386.SYS device driver line in CONFIG.SYS).

If you have programs (e.g., DESQview, DESQview/X) that can take advantage of additional High RAM, you may want to enable Stealth ROM even if Optimize did not recommend Stealth ROM testing—this will give DESQview and DESQview/X users more memory for each window.

If you do not know if you are using Stealth ROM, you can easily find out.

- **At the DOS prompt, type QEMM and press Enter ↵.**

  A report summarizing QEMM's status will display. If you see the line **Stealth Type = M** or **Stealth Type = F**, Stealth ROM is already enabled on your system.

  If you are not using Stealth ROM, you can enable it as follows:

- **Be sure you are at the DOS prompt, outside of any multitasking program such as DESQview, DESQview/X, or Microsoft Windows. Type OPTIMIZE /STEALTH and press Enter ↵, then follow the on-screen instructions.**

  Optimize will first try the Stealth ROM mapping method (also called ST:M) and, if it is appropriate for your system, Optimize will enable that method. If the mapping method is not right for your system, Optimize will try the frame method (also called ST:F).

Disabling Stealth ROM

If you are using Stealth ROM and would like to disable it:

- **Edit your CONFIG.SYS file and delete the ST:F or ST:M parameter from the QEMM386.SYS device driver line.**

  If you have added any Stealth ROM related parameters to the QEM386.SYS line, be sure to remove them along with the ST:M or ST:F parameter.

- **Run Optimize (see Chapter 3).**

A Technical Description of Stealth ROM

Stealth ROM creates additional mappable areas at the addresses used by your PC's ROMs. By default, QEMM will turn these areas into High RAM that can be used to load TSRs, device drivers and selected parts of DOS. When Stealth ROM is enabled, QEMM monitors the interrupts pointing into those ROMs. When those interrupts occur, QEMM maps the appropriate ROM into the page frame and passes the interrupts to the ROM's location in the page frame. In general, the ROMs targeted are your system (BIOS) ROM, video ROM and disk ROM (certain other ROMs may be "Stealthed" as well).

With the ROMs "out of the way," the amount of usable upper memory is greatly increased. Depending on the location of the ROMs,

High RAM regions can become quite large and able to accommodate more or larger device drivers and TSRs.

Because of differences in the way that hardware and software deal with ROMs, Stealth ROM can be implemented using one of two methods:

- **ST:M** (the mapping method) maps system, video, and disk ROMs and any other "Stealthable" ROMs out of the first megabyte of memory. When the system needs the ROM, QEMM maps the appropriate ROM code into the EMS page frame. The ROM code then has a valid real mode address at which it can execute, and it does so normally. When the ROM routine is finished, QEMM restores the previous contents of the page frame.
- **ST:F** (the frame method) leaves the system, video, and disk ROMs in place. QEMM places the EMS page frame so that it lies on top of a ROM's address space. When the ROM at the location of the page frame is needed, QEMM saves the current contents of the page frame and restores the ROM to its original location. The ROM code then executes normally. When the ROM routine is finished, QEMM restores the previous contents of the page frame.

The mapping method can typically provide 83K-115K of extra High RAM, and the frame method, which is compatible with more systems, typically provides 48K-64K of extra High RAM.

Stealth ROM is enabled by adding the parameter ST:M or ST:F to the QEMM386.SYS device driver line in CONFIG.SYS. This is normally done for you by the Optimize program. For more information on Stealth ROM, see the STEALTHROM parameter in **Chapter 7**.

Stealthing DOS 6's DoubleSpace Driver

MS-DOS 6 includes a program you can use to compress the data on your hard drive and floppy disks so they will hold more data.

*The name of DOS's disk compression feature depends on which version of DOS you have. In DOS versions 6.0 through 6.20, it is called DoubleSpace; starting with DOS version 6.22 Microsoft calls the feature DriveSpace (version 6.21 does not have a disk compression utility). QEMM's Stealth D*Space feature will work with both DoubleSpace and DriveSpace.*

DoubleSpace and DriveSpace use a device driver that takes up 34K to 52K of conventional or upper memory, depending on the DOS version and the options you use. Without QEMM, DOS will load that driver into conventional or upper memory. If you load the driver into conventional memory, you have 34-52K less room for running programs; if you load it into upper memory, you have 34-52K less space for loading TSRs and device drivers high.

QEMM has a feature called Stealth D*Space (pronounced "Stealth Dee-space") that moves the DoubleSpace or DriveSpace driver out of conventional or upper memory and maps it into the EMS page frame whenever it is needed. The Stealth D*Space feature saves you about 31K to 49K of memory, depending on your configuration.

*In QEMM versions 7.0-7.04, Stealth D*Space was known as Stealth DoubleSpace and its driver, which is now called ST-DSPC.SYS, was called ST-DBL.SYS*

Enabling or Disabling Stealth D*Space

When you install QEMM, the installation program will detect if you are using DoubleSpace or DriveSpace and will offer to enable the Stealth D*Space feature. You may want to enable Stealth D*Space if you used DoubleSpace or DriveSpace to compress your hard drive *after* installing QEMM.

You can enable or disable Stealth D*Space form the QEMM Setup program. We will explain how to do this in the DOS version of QEMM Setup, though you could use the Windows version if you like (for information on using the Windows version, see **Chapter 2**). To enable or disable Stealth D*Space:

Start QEMM's Setup program.

- **To start Setup from DOS, type SETUP and press Enter ↵.**
- **To start Setup from Microsoft Windows, go to the QEMM group and select QEMM Setup.**
- **Press Enter ↵ to go to the Setup menu.**
- **Select Enable or disable Stealth D*Space, and when you are asked if you want to enable Stealth D*Space, select Yes to enable it, or No to disable it..**
- **Run Optimize (see Chapter 3).**

*You do not have to have Stealth ROM itself enabled to use Stealth D*Space. Stealth D*Space uses the EMS page frame—if you want to use Stealth D*Space, you must do nothing that would eliminate or reduce the size of the page frame (i.e., do not use any of the following parameters on the QEMM386.SYS device driver line in CONFIG.SYS: EMS:N; FRAMELENGTH=0, 1, 2 or 3; or FRAME=NONE).*

Improving Disk Performance when using Stealth D*Space

There can be potential conflicts if QEMM's Stealth D*Space and Stealth ROM features are both enabled. To prevent such conflicts, Stealth D*Space automatically creates a 1K disk buffer on systems where the Stealth ROM feature is active. Stealth D*Space does not automatically create this buffer if QEMM is already doing its own buffering (see the DISKBUF and DISKBUFFRAME parameters in **Chapter 7**).

Stealth D*Space's disk buffer can slow down disk access time. If your disk performance seems slower, you may be able to improve it by increasing the size of the disk buffer. A larger disk buffer will use more conventional memory or High RAM (depending on where ST-DSPC.SYS is loaded). To enlarge the Stealth D*Space disk buffer, edit your CONFIG.SYS file and add the DB:x parameter at the end of the ST-DSPC.SYS device line, where x is the desired buffer size in K. A buffer size of 4K should virtually eliminate speed degradation (at the expense of 3 additional K of memory).

You can also force Stealth D*Space to create or not to create its disk buffer by using the parameter DB:Y (to create the buffer) or DB:N (not to create the buffer) on the ST-DSPC.SYS device line. There is normally no reason to use DB:Y; ST-DSPC.SYS should automatically create the buffer if needed. If you do not need the buffer, you can speed up disk access time and free up 1K of memory by using the DB:N parameter. DB:N can be useful in the following circumstances:

- Some disk caches make Stealth D*Space's disk buffering unnecessary. If you are using a disk cache, you might use DB:N if you do not access your compressed drive before the cache is loaded. If you load CONFIG.SYS and AUTOEXEC.BAT from the compressed drive, you should not use DB:N because the D*Space drive will probably be accessed before the cache is loaded.

- If you are using Stealth ROM, but hard disk functions are not being Stealthed, you can use DB:N. This may be the case if you use the QEMM386.SYS parameter XSTI=13, or if you use the XST

parameter on the ROM that manages hard disk access. These parameters are only for troubleshooting problems related to Stealth ROM; do not use them just to be able to use ST-DSPC.SYS's DB:N parameter.

You can configure some disk caches to use expanded memory. If Stealth D*Space detects a disk cache using expanded memory, it will automatically create a 1K disk buffer specifically designed to make Stealth D*Space work with EMS-using disk caches. If you have an EMS-using cache that fails when it starts, or your system freezes when ST-DSPC.SYS loads, Stealth D*Space may not be detecting the EMS-using disk cache. In this case, you should force Stealth D*Space to create the necessary 1K disk buffer by adding the parameter EXPCACHE:Y to the ST-DSPC.SYS device line in CONFIG.SYS.

Because the disk buffer may cause a slowdown in disk access time, you can create a larger disk buffer to speed up disk access time by using EXPCACHE:x, where x is the desired buffer size in K. A larger disk buffer will use more conventional memory or High RAM (depending on where ST-DSPC.SYS is loaded). A buffer size of 4K should virtually eliminate speed degradation (at the expense of 3 additional K of memory). If you have some need to force Stealth D*Space not to create the disk buffer, you can use the parameter EXPCACHE:N.

A Technical Description of Stealth D*Space

The name of DOS's disk compression drivers depend on whether you are using DoubleSpace or DriveSpace. DoubleSpace's drivers are DBLSPACE.BIN and DBLSPACE.SYS. DriveSpace's drivers are DRVSPACE.BIN and DRVSPACE.SYS.

DBLSPACE.BIN/DRVSPACE.BIN is the part of DOS that provides access to your compressed drive. If you are using DoubleSpace/DriveSpace, DOS loads DBLSPACE.BIN/DRVSPACE.BIN before processing the commands in CONFIG.SYS. When you run DoubleSpace's setup, it adds a device command for DBLSPACE.SYS/DRVSPACE.SYS to your CONFIG.SYS file. DBLSPACE.SYS/DRVSPACE.SYS is actually a device driver that moves DBLSPACE.BIN/DRVSPACE.BIN into conventional memory.

QEMM's Stealth D*Space feature moves the DBLSPACE/DRVSPACE driver outside of the first megabyte of memory, freeing up the memory DBLSPACE/DRVSPACE originally

occupied. QEMM hooks all DBLSPACE's/DRVSPACE's entry points and maps DBLSPACE/DRVSPACE into the page frame when any program uses one of the DBLSPACE/DRVSPACE driver's entry points.

To load Stealth D*Space, the DBLSPACE.SYS/DRVSPACE.SYS line in CONFIG.SYS should be replaced with the following:

DEVICE=C:\QEMM\ST-DSPC.SYS

Stealth D*Space leaves a 3K stub in conventional memory. When you run Optimize after enabling Stealth D*Space, Optimize will attach the necessary LOADHI statement to the beginning of this line to load the 3K stub high.

NOTES:

6 VIDRAM:
Extending Conventional Memory

QEMM's VIDRAM program can extend conventional memory by as much as 96K for running DOS text-based programs. VIDRAM even extends conventional memory for DOS programs running in Microsoft Windows. You may want to read this chapter if you routinely run large DOS-text based programs (e.g., dBASE IV) that could benefit from the additional memory.

To use VIDRAM, your system must have an EGA or VGA video adapter or an adapter with EGA or VGA capability (this includes VGA-compatible 8514 video adapters). Your PC must have 640K of conventional memory and the programs you run using this additional memory must not use EGA or VGA graphics.

VIDRAM has one other requirement: For VIDRAM to work, no hardware device (e.g., a hard disk controller) can utilize the area just below 640K. QEMM will automatically move XBDAs (Extended BIOS Data Areas) out of this area.

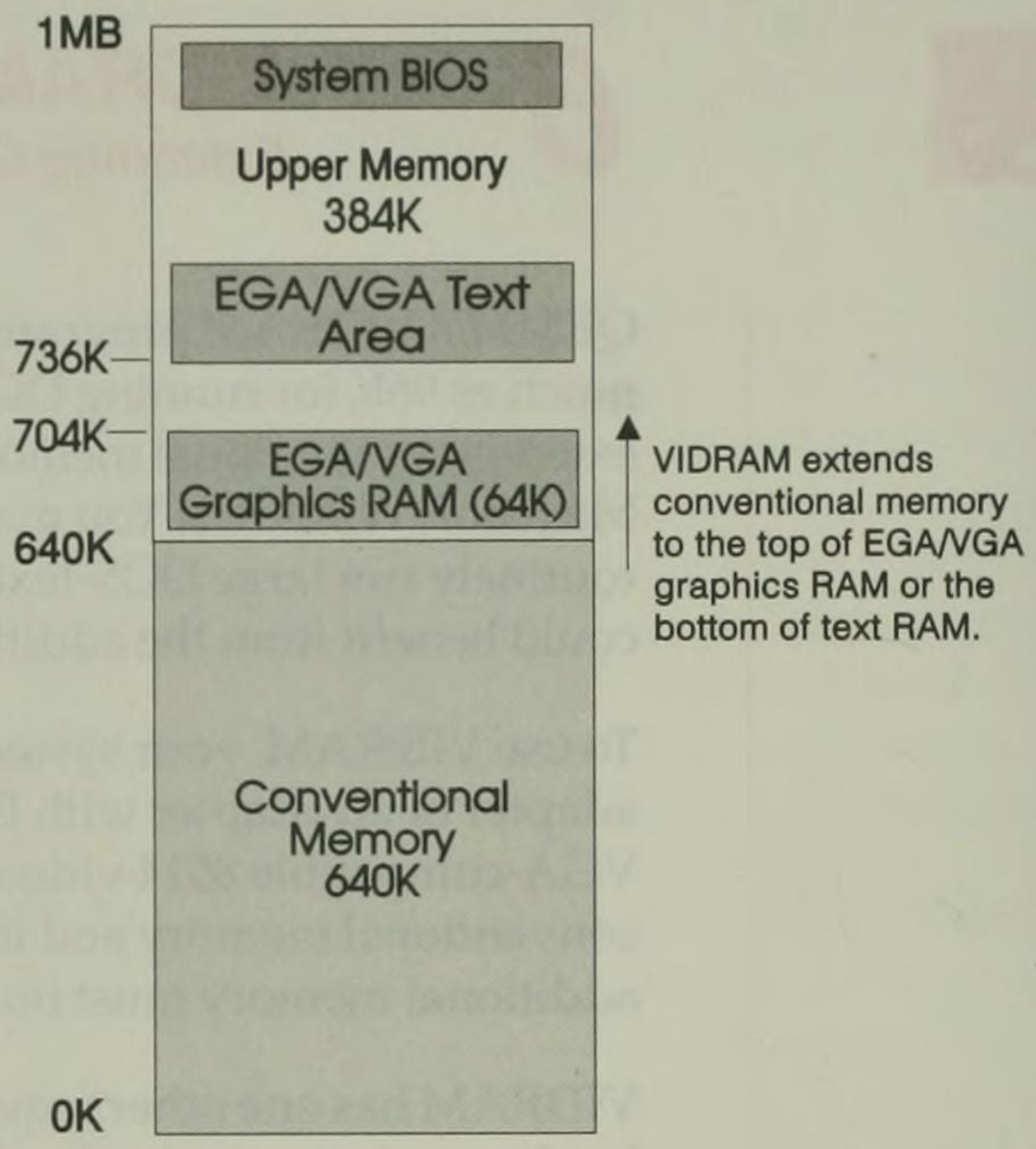
VIDRAM is a standalone TSR. It does not require expanded memory, extended memory or a memory manager. If your system has the necessary video RAM, VIDRAM will work on 8088, 8086, 80286 and 80386, i486 and Pentium PCs.

If your system has a Hercules, monochrome or CGA video adapter (and it does not have an additional EGA or VGA adapter), you will not need VIDRAM. QEMM automatically extends conventional memory to 704K on Hercules and monochrome systems and to 736K on CGA systems.

How VIDRAM Works

If your PC has an EGA or VGA video adapter, the 64K memory area just above conventional memory (640K-704K, or A000-AFFF in hex format) is reserved for use by graphics modes. When you run text-based programs, that area is unused, so VIDRAM can appropriate it to extend the contiguous conventional memory for running programs (see the illustration on the following page).

There may be an additional 32K of unused address space directly above the video graphics area that VIDRAM can also appropriate. As



Memory areas used by VIDRAM

long as the memory addresses are contiguous with the end of conventional memory and they are in the first megabyte, DOS can use them to run real mode programs.

If your PC has a VGA-compatible 8514A adapter: You can use VIDRAM to extend conventional memory for 8514A graphics programs as well as DOS text-based programs. Graphics programs configured specifically for the 8514A card do not use the EGA/VGA graphics area, so you will still be able to use these programs when VIDRAM is on.

VIDRAM actually does two things:

- It intercepts video requests and refuses all requests that would make use of the EGA/VGA graphics area (i.e., graphics operations).
- It maps EMS memory (obtained via a private interface to QEMM) to the EGA/VGA area, unless you specify otherwise. If you do not want to use EMS memory, you can tell VIDRAM to use your video card's memory, which is generally slower than EMS memory.

The amount of additional DOS memory VIDRAM will make available depends on several factors. The maximum is 96K with a color monitor and an EGA or VGA adapter. If you have a monochrome monitor or are using dual monitors you can still gain 64K of memory for your programs.

Using VIDRAM

It is important to understand that you cannot run EGA or VGA graphics operations while VIDRAM is in use. If you routinely use both large text-based programs and graphics programs, you can turn the VIDRAM feature on when you need it for a text program and off before you run a graphics program. If you are using an 8514A adapter, you can still use 8514A graphics programs while VIDRAM is enabled.

To turn VIDRAM on:

- **Type VIDRAM ON and press Enter ↵.**

This command will extend conventional memory into the EGA/VGA graphics area for a total of 704K conventional memory.

WARNING: If VIDRAM is on and you start an EGA or VGA graphics program, you will see a VIDRAM warning, telling you that graphics video modes are disabled. At that point, you should press any key other than Esc to abort the graphics program and return to the DOS prompt. Then, turn VIDRAM off and you can start your graphics program again.

To turn VIDRAM off:

- **Type VIDRAM OFF and press Enter ↵.**

If you have a color system and you want to extend conventional memory an additional 32K for a total of 736K, you will have to prevent QEMM from using the 32K directly above the EGA/VGA graphics area as High RAM. Keep in mind that there will be 32K less space to load TSRs or device drivers high, so you will need to decide whether having the extra 32K for DOS text programs outweighs the loss of High RAM. If you decide to extend conventional memory by 32K, add the VIDRAMEGA parameter to the QEMM386.SYS device driver line in your CONFIG.SYS file, reboot your PC, then type **VIDRAM ON**. See **Chapter 7** for information on the VIDRAMEGA parameter.

DESQview users: If you want to extend conventional memory for DESQview DOS windows, you must first add the VIDRAMEMS parameter to the QEMM386.SYS device driver line in your CONFIG.SYS file, then run Optimize (see **Chapter 7** for information on the VIDRAMEMS parameter and **Chapter 3** for information on Optimize). Then, before you start DESQview, type **VIDRAM ON.** This will ensure that the additional memory is available to all DOS windows in DESQview. You cannot turn VIDRAM on and off inside DESQview.

DESQview/X users: Because DESQview/X is a graphics program and VIDRAM uses the EGA/VGA graphics area, you cannot use VIDRAM with DESQview/X on an EGA/VGA system. If you have an 8514A video adapter and are running DESQview/X in 8514A mode, you can use VIDRAM to extend your DOS windows under DESQview/X; just follow the instructions in the preceding paragraph. When DESQview/X starts, you may see the message, "Graphics driver: GRFVGA does not find the correct video adapter." Ignore that message and do not be concerned if DESQview/X's title screen displays in text mode instead of graphics mode. Remember, you will not be able to run any EGA/VGA graphics programs when VIDRAM is on.

If you seldom use graphics programs, you may want to put VIDRAM in your AUTOEXEC.BAT file so it will load whenever you boot your PC. To do this, add the line **VIDRAM ON** to your AUTOEXEC.BAT file.

If you ever want to know whether VIDRAM is on or off:

- **Type VIDRAM and press Enter ↵.**

You will see a message telling you whether VIDRAM is resident and enabled. If it is enabled, it will report the type of memory in use and the new ending address of conventional memory.

VIDRAM Parameters

VIDRAM has several parameters. These parameters will be useful if:

- You need to run graphics programs occasionally.
- You have a second video adapter and monitor.

VIDRAM's optional parameters are listed below. Some parameters have an abbreviation you can use instead of the full name; abbreviations are listed in parentheses.

- **RESIDENT (RES)**

  loads the resident portion of VIDRAM into memory, but does not enable any VIDRAM functions. This parameter is provided for those who need to load VIDRAM high. If you want to use this parameter, see the next section, "Loading VIDRAM High."

- **OFF (OF)**

  turns VIDRAM off, restoring video memory to the reserved video area and enabling graphics capability.

- **ON**

  turns VIDRAM on to extend DOS memory and inhibit graphics operations. VIDRAM will remain on until you turn it off. You can modify the ON parameter with three additional options: OVERRIDE, EGA and EMS.

  - **VIDRAM ON OVERRIDE (OV)** forces VIDRAM on in situations where it would normally refuse to appropriate video memory. These situations occur when you have two monitors and two video adapters in your system, or when all or part of the video RAM is already being managed by a resident memory manager. If either of these situations applies to you, we suggest you try VIDRAM ON OVERRIDE to see if it helps. OVERRIDE's utility in these situations depends on what adapters are involved or what memory management features you are using.

  - **VIDRAM ON EGA** tells VIDRAM to use the memory supplied by the video adapter instead of EMS memory. You can use this option when you want to preserve every bit of EMS memory for other uses. Video adapter memory is generally slower than EMS memory. This is the default behavior when there is not enough EMS memory available or QEMM is not present. If you use VIDRAM ON EGA and you want to extend conventional memory an extra 32K, you can add the VIDRAMEGA parameter on the QEMM386.SYS device driver line of your CONFIG.SYS file (see **Chapter 7** for information). If you add that parameter, you will forfeit 32K of High RAM. If you add the parameter, be sure to run Optimize (see **Chapter 3**).

- **VIDRAM ON EMS** tells VIDRAM to use EMS memory (provided by the standard EMS interface) to extend DOS. This parameter is not necessary when you are using QEMM. You can use VIDRAM ON EMS if you want VIDRAM to use EMS memory and you are using another memory manager that can make the video graphics area mappable.

- **NOCGA**

  This parameter is rarely used. It prevents all graphics functions (CGA through VGA) from using the video graphics areas. This option makes VIDRAM resident but does not extend DOS memory. You might use this parameter when some of your video RAM area has been mapped with EMS memory by something other than VIDRAM (e.g., QEMM386.SYS's INCLUDE=A000-AFFF parameter) and you want to protect it from being accessed by graphics programs.

- **NOEGA**

  This parameter is rarely used. It prevents EGA and VGA graphics requests from being honored. This option makes VIDRAM resident but does not extend DOS memory. You might use this parameter when some of your video RAM area has been mapped with EMS memory by something other than VIDRAM (e.g., QEMM386.SYS's INCLUDE=A000-AFFF parameter) and you want to protect it from being accessed by graphics programs.

Loading VIDRAM High

If you are using VIDRAM, you can save a small amount of memory by loading the resident portion of VIDRAM high. Once you load VIDRAM high, you will need to type a second VIDRAM command to extend conventional memory. To load VIDRAM's resident portion high and extend conventional memory into the VGA graphics area:

- Type **LOADHI VIDRAM RESIDENT** and press **Enter** ↵
- Type **VIDRAM ON** and press **Enter** ↵.

  This allows the code that intercepts graphics requests to be in High RAM, while the memory management portion is in memory only briefly to extend or return memory. The resident portion of VIDRAM is quite small.

Using VIDRAM with Microsoft Windows

You can use VIDRAM with Microsoft Windows 3.0 and 3.1 386 enhanced mode to extend conventional memory for running DOS text-based applications under Windows.

IMPORTANT: If VIDRAM is on before you start Windows, turn it off by typing VIDRAM OFF.

To use VIDRAM to extend conventional memory for a DOS text-based program in Windows:

- **Start Windows. Be sure you are running in 386 enhanced mode. You can use WIN /3 to ensure that you start in that mode.**
- **Double-click on the MS-DOS Prompt icon (usually in the Main group).**
- **When you see the DOS prompt, type VIDRAM ON and press Enter ↵.**
- **If you want to run the DOS application in a window, press Alt+Enter ↵.**
- **Type the command to start the program.**
- **When you close the window, the extra memory that VIDRAM provides will be automatically relinquished.**

  If you want to use VIDRAM every time you run a certain DOS text program under Windows, you could create a batch file containing two lines: VIDRAM ON and the command to start the program. Then you could use Windows PIF editor to put the batch file's name in the PIF's Program Filename field (see your Windows manual for information).

If you want to extend conventional memory an additional 32K (for a total of 736K) for DOS text programs running in Windows, add the parameter VIDRAMEGA to the QEMM386.SYS device driver line in your CONFIG.SYS file (see **Chapter 7***). Keep in mind that there will be 32K less space to load TSRs or device drivers high, so you will need to decide whether having the extra 32K for DOS text programs outweighs the loss of High RAM.*

NOTES:

Technical Reference

7 QEMM386.SYS Parameters

This chapter is the technical reference guide for the parameters that can be used with the QEMM386.SYS device driver in your CONFIG.SYS file. Since QEMM automatically configures itself for your system, you should only need to use the parameters in this chapter if you want to fine-tune your memory configuration, or you are experiencing problems.

To use a QEMM command line parameter, place the parameter name on the same line as **DEVICE=QEMM386.SYS** in your CONFIG.SYS file. For example, the QEMM command line with the RAM, ROM=C000, and STEALTHROM:M parameters set looks like:

DEVICE=QEMM386.SYS RAM ROM=C000 STEALTHROM:M

IMPORTANT: Do not put spaces within a parameter. In the example above, make sure that you do not have a space before or after the = sign in ROM=C000. Also, any parameters you use must appear on the same line as **DEVICE=QEMM386.SYS.** *Optionally, you can put the parameters you want to use in a file and reference the file on the QEMM386.SYS line (see the section "Parameter Files," later in this chapter).*

If you have upgraded from QEMM 6 or earlier, be aware that many of the parameter names have changed. You can still use the older parameter names if you like. At the end of this chapter is a list of the parameter names that have changed, cross-referenced with their former names.

Parameters and Memory Addresses

All memory addresses in this chapter that are used as arguments to QEMM parameters are specified as hexadecimal memory addresses. Even if you do not understand hexadecimal addresses, you should be able to use these parameters; you can generally get the appropriate values from either the documentation for your hardware or software, or from the QEMM program (e.g.,Manifest, Optimize or QEMM.COM) that suggests the parameter.

QEMM's parameters also accept decimal numbers followed by the letter K, to denote a number of kilobytes, or the number M, to denote

a number of megabytes. So the parameter **EXCLUDE=B000-B7FF**, where **B000** and **B7FF** are hexadecimal segment addresses, can also be specified as **EXCLUDE=704K-736K**. You can mix different kinds of numbers in the same parameter (e.g., **ROM=896K-1M**).

Furthermore, you can specify an address range either with a hyphen (–), which indicates that the number following the range's starting point is its end point, or with a colon (:), which indicates that the number following the range's starting point is its length. For example, the parameter **RAM=C000-C0FF** can also be specified as **RAM=C000:4K**, or as **RAM=768K:4K**.

Parameter Files

If you do not want to place some or all of the parameters you desire on the QEMM386.SYS line in your CONFIG.SYS file, you can tell QEMM to read parameters from a text file. You can specify a full pathname to the text file, or you can specify only its name, and place it in the directory that is current when QEMM loads (usually the root directory of your boot drive). To use a parameter file, place a @ character on the QEMM line, immediately followed by the filename. For example, the following line tells QEMM to read parameters from a file called SWITCHES.CMD in the root directory:

```
DEVICE=C:\QEMM\QEMM386.SYS RAM @SWITCHES.CMD
```

Note that some parameters can be specified directly on the QEMM line and others in the SWITCHES.CMD file. The parameter file can contain a separate parameter on each line, or you can place all the parameters on the same line with a space between each parameter. A semicolon in a parameter file means that everything to the right of the semicolon on that line is a comment and will not be processed by QEMM. So SWITCHES.CMD might look something like this:

```
ROM ;Run ROMs fast
DB=10 ;Remove when VDS driver arrives
ST:M ;Stealth
```

You can specify parameter files from within parameter files, down to five levels. A parameter file could look like this:

```
RAM
@EXCLUDES.CMD
ST:M
ROM
```

In this case, a separate file in the root directory called EXCLUDES.CMD would contain additional parameters for QEMM to process.

Parameter Descriptions

Many of the parameters have an abbreviation you can use instead of the full name; abbreviations are listed in parentheses below.

Several parameters take a Y or N argument (for "yes" or "no"). Even though these parameters can take either Y or N as an argument, we often list only one of these arguments (e.g., **SORT:Y** or **USEXMS:N**) because the other is QEMM's default condition at all times. If the parameter's default varies depending on other conditions (e.g., whether or not Stealth ROM is enabled), then we describe all the parameter's arguments. If you specify neither a Y nor an N to such a parameter, QEMM will act as if you specified a Y.

Often a parameter has a certain effect when used along with another parameter, so we have added comments to describe the interactions and relations between parameters. QEMM processes its parameters sequentially—if two mutually exclusive parameters are placed on the QEMM386.SYS line in the CONFIG.SYS file, the one that comes later overrides the earlier one.

Commonly Used Parameters

The most commonly used parameters are INCLUDE, RAM, ROM, STEALTHROM and EXCLUDE. The first four of these parameters activate QEMM features, and the last one is the most useful troubleshooting parameter. These five parameters are described below in alphabetical order.

- **EXCLUDE=xxxx-yyyy (X=xxxx-yyyy)**
  tells QEMM to make the region **xxxx-yyyy** unavailable for High RAM, expanded memory mapping or ROM mapping. EXCLUDE is the most important troubleshooting parameter, and is usually used to stop QEMM from placing High RAM in a section of upper memory that a program or a piece of hardware needs to access. For example, if you have a network adapter that puts 16K of RAM at address CC00, and QEMM is not detecting the network adapter, then you should use the parameter EXCLUDE=CC00-CFFF. You can use the Analysis procedure (see *page 125*) to find regions that should be excluded.

You may specify multiple ranges by using EXCLUDE several times. If you exclude any part of a 4K region aligned on a 4K boundary, QEMM will exclude that entire 4K region.

By default, QEMM will try to reserve 64K of contiguous upper memory addresses that are not used by ROMs, video RAM or adapter RAM, and designate this area as the EMS page frame. When you use the EXCLUDE parameter, you reduce the number of possible locations for the page frame. If QEMM cannot find 64K of free contiguous addresses in upper memory, it places the page frame in conventional memory, diminishing by 64K the amount of memory available to run programs.

You may find it convenient to use one of the other ways of specifying an address range. For example, to exclude a 16K area starting at CC00, you could use EXCLUDE=CC00:16K. See "Parameters and Memory Addresses" on ***page 69*** *for more information.*

- **INCLUDE=xxxx-yyyy (I=xxxx-yyyy)**
  tells QEMM to use the address range **xxxx-yyyy** for High RAM or for expanded memory mapping. If you include an address range, QEMM will use it regardless of whether it is used by ROMs, video RAM, or adapter RAM. We advise you to use the INCLUDE parameter only when recommended by the Analysis procedure (see *page 125*).

  Used properly, the INCLUDE parameter may increase the amount of available High RAM, particularly on systems that are not using the Stealth ROM feature. You can specify multiple regions by using INCLUDE several times. INCLUDE operates only on 4K regions aligned on 4K boundaries; QEMM will not include any part of a 4K region aligned on a 4K boundary unless the entire region is included.

You may find it convenient to use one of the other ways of specifying an address range. For example, to include a 4K area starting at F100, you could use INCLUDE=F100:4K. See "Parameters and Memory Addresses" on ***page 69*** *for more information.*

- **RAM or RAM=xxxx-yyyy**
  specifies that QEMM should create High RAM—Quarterdeck's term for RAM that QEMM makes appear in upper memory (the area

between 640K and 1024K). You need High RAM if you want to load TSRs, device drivers or portions of DOS high, to free up more conventional memory for running DOS programs. By default, the QEMM installation adds the RAM parameter to the QEMM386.SYS line.

If you specify the RAM parameter without an address range, QEMM sets aside room for an EMS page frame (unless you are using the EMS:N, FRAME=NONE or FRAMELENGTH=0 parameter) and places High RAM at all other upper memory addresses at which QEMM has not detected ROM, video RAM or adapter RAM. Do not specify an address range unless you know what regions of upper memory are safe to use.

High RAM can be used by QEMM's LOADHI programs, by DESQview and DESQview/X, and by programs that allocate UMBs (Upper Memory Blocks) through the XMS specification. When the RAM parameter is specified, you cannot turn QEMM off.

- **ROM, ROM=xxxx or ROM=xxxx-yyyy**

  tells QEMM to speed up the operation of ROMs by copying them into faster RAM and making the RAM appear in upper memory in place of the ROMs. This process is called *ROM shadowing*. If your system does not shadow ROMs in hardware, the ROM parameter may speed up some system operations. It particularly affects programs that write to the screen using BIOS or DOS video calls (like DOS's COMMAND.COM). If your system shadows ROMs in hardware and you have this feature enabled in your system setup, you normally would not need the ROM parameter.

  If you specify ROM without an address, QEMM copies all ROMs into RAM. If you specify only the starting address of a ROM, QEMM will determine the ROM's size, and shadow that entire ROM and no other. For example, ROM=C000 will make QEMM shadow a video ROM that begins at C000 and will leave all other ROMs unaffected. You can specify both the beginning and end of an address range if you want QEMM to shadow only part of a ROM, but it is usually easier to use the EXCLUDE parameter to tell QEMM what areas should not be affected by the ROM parameter. For instance, if you want to shadow all ROMs, but you know that shadowing the F400-F4FF region causes your floppy drives to malfunction, you can use the parameter ROM EXCLUDE=F400-F4FF.

When the ROM parameter is in use, you cannot turn QEMM off. The smallest region that the ROM parameter can affect is a 4K region aligned on a 4K boundary, and specifying any part of such a region causes the whole 4K region to be shadowed. However, when a ROM has been relocated with the STEALTHROM parameter, the smallest region that the ROM parameter can affect is a 16K region aligned on a 16K boundary. In this case, the 16K region will not be shadowed unless a portion of each of its 4K regions has been specified. For example, if the C000 ROM has been relocated by STEALTHROM, ROM=C000-C2FF will have no effect, but ROM=C000-C37F will cause the entire C000-C3FF region to be shadowed.

If you use the EXCLUDE parameter to prevent a ROM region from being shadowed by the ROM parameter, then any EXCLUDE will normally prevent the entire 4K ROM region (aligned on a 4K boundary) in which it falls from being shadowed. However, if that ROM is being relocated with the STEALTHROM parameter, then any EXCLUDE, however small, will prevent the entire 16K ROM region (aligned on a 16K boundary) in which it falls from being shadowed. For more information on using Stealth ROM with the ROM parameter, see "Using the ROM and STEALTHROM Parameters Together" on *page 102*.

- **STEALTHROM:M or F (ST:M or F)**

  enables the Stealth ROM feature—the QEMM technology that relocates ROMs from upper memory so that their addresses can be used for High RAM or for expanded memory mapping. When a ROM service is requested, QEMM will temporarily place the ROM in the EMS page frame to handle the request. Stealth ROM is an important feature—with it, QEMM can typically create an additional 48K-115K of High RAM for loading TSRs, device drivers, and portions of DOS.

  When you specify the STEALTHROM:x parameter, replace the letter **x** with the letter that designates the Stealth ROM method that you want to use. **M** specifies the **mapping method**, which relocates as many ROMs as possible out of upper memory. **F** specifies the **frame method**, which relocates only the ROMs in the area where the EMS page frame lies. The mapping method usually makes more upper memory available, but the frame method may work in some cases when the mapping method does not. On a typical system, the mapping method frees an additional 83K-115K, and the frame method frees 48K-64K.

Depending on how your system is configured, QEMM's Optimize process may test your system for Stealth ROM compatibility and specify a Stealth ROM parameter automatically. If you want to test for Stealth ROM compatibility, run Optimize with the /STEALTH parameter (see **Chapter 3**).

Using the EXCLUDE parameter in a ROM region does not prevent QEMM from relocating the ROM when the STEALTHROM parameter is in use. However, it does prevent QEMM from placing High RAM or allowing memory to be mapped in that region. When the EXCLUDE parameter is used on an area from which ROMs have been moved, programs are able to access the ROM at its original location, even though all ROM BIOS calls are still redirected away from this region by the Stealth ROM process. This is sometimes the most efficient way to preserve the benefits of Stealth ROM when you are running a program that insists on accessing a ROM at a fixed address. For instance, the parameters STEALTHROM:M EXCLUDE=FC00-FCFF allow you to use the address space of all Stealthed ROMs with the exception of the 4K region at FC00-FCFF, in the event that you have a program that jumps directly to that location to access a ROM routine.

For information on using Stealth ROM with the ROM parameter, see "Using the ROM and STEALTHROM Parameters Together" on *page 102*. See also the EXCLUDESTEALTH and EXCLUDESTEALTHINT parameters for discussions of how to modify the Stealth ROM feature. You cannot use Stealth ROM if you have specified the EMS:N, FRAME=NONE, or FRAMELENGTH=0 parameter.

Less Frequently Used Parameters

The previous section of this chapter described QEMM's most commonly used parameters. This section describes the rest of QEMM's parameters. The parameters are described in alphabetical order.

- **?**

  displays a list of the QEMM386.SYS command line parameters and their abbreviations. QEMM will not load if you specify this parameter.

- **ADAPTERRAM=xxxx-yyyy (ARAM)**

  tells QEMM to make the region **xxxx-yyyy** unavailable for High RAM, expanded memory mapping or ROM mapping.

ADAPTERRAM is exactly like the EXCLUDE parameter, except that the QEMM.COM Type display and the Manifest QEMM Type display will identify this region as "Adapter RAM" instead of "Excluded." Like the EXCLUDE parameter, it is needed only if QEMM does not identify an adapter's RAM.

You may find it convenient to use one of the other ways of specifying an address range. For example, to exclude a 16K area starting at D000, you could use ARAM=D000:16K. See "Parameters and Memory Addresses" on ***page 69*** *for more information.*

- **ADAPTERROM=xxxx-yyyy (AROM)**

  tells QEMM to make the region **xxxx-yyyy** unavailable for High RAM, expanded memory mapping or ROM mapping. ADAPTERROM is exactly like the EXCLUDE parameter, except that the QEMM.COM Type display and the Manifest QEMM Type display will identify this region as "Adapter ROM" instead of "Excluded." Like the EXCLUDE parameter, it is needed only if QEMM does not identify an adapter's ROM.

You may find it convenient to use one of the other ways of specifying an address range. For example, to exclude a 16K area starting at C800, you could use AROM=C800:16K. See "Parameters and Memory Addresses" on ***page 69*** *for more information.*

- **AUTO (AU) or ON or OFF (OF)**

  determines when QEMM puts the processor into virtual-8086 mode, which is required for most of QEMM's functions. These parameters are generally used only for troubleshooting purposes.

  AUTO means that QEMM will put the system into virtual-8086 mode only when necessary to provide a feature, like expanded memory. ON means that QEMM will always run real-mode programs in virtual-8086 mode. OFF means that QEMM will never use virtual-8086 mode; QEMM will then provide almost no features, though it will still take extended memory and make it unavailable to other programs. (You can use the EXTMEM or MEMORY parameters to limit QEMM's memory allocation even when QEMM is off.)

  Various other parameters (e.g., RAM, ROM, STEALTHROM) force QEMM on and cause AUTO and OFF to be ignored. If no other

parameter forces QEMM on, QEMM's state defaults to AUTO, but can be changed from the DOS prompt with the QEMM.COM program (see **Chapter 9**).

- **BOOTENABLE:N (BE:N)**

  tells QEMM to disable its QuickBoot feature, which allows you to warm boot the system quickly without using the BIOS's reboot process. You can use BOOTENABLE:N if you have compatibility problems with QuickBoot that are not fixed by the other QuickBoot-related parameters (BOOTFLOPPY:N, BOOTTIMEOUT=xx, and BOOTVIA19:Y).

- **BOOTFLOPPY:N (BF:N)**

  tells QEMM to bypass the floppy drives when it is warm booting the system with its QuickBoot feature. When you use BOOTFLOPPY:N, you cannot boot your system with a boot diskette in the floppy drive. Without this parameter, QEMM emulates the BIOS's boot process, checking the A: floppy drive first and the C: drive next for a bootable disk.

  If you are using the BOOTTIMEOUT=xx parameter to post a QuickBoot menu during the warm boot, BOOTFLOPPY:N changes the default choice on the QuickBoot menu to "Boot from Drive C:" instead of the usual default of "Boot from Drive A:". In this configuration BOOTFLOPPY:N still tells QEMM to bypass the floppy drive by default, but it does not prevent you from booting from the floppy drive if you choose the "Boot from Drive A:" menu option.

- **BOOTTIMEOUT=xx (BTO=xx)**

  tells QEMM to post a QuickBoot menu for xx seconds when you warm boot your system. The QuickBoot menu lets you choose which drive to boot from; when the timeout value of xx seconds expires, QEMM reboots the system without your intervention. BOOTTIMEOUT=0 is the default, and tells QEMM to warm boot without posting the Quickboot menu. See the BOOTFLOPPY:N parameter for information on how to change the default option on the QuickBoot menu.

- **BOOTVIA19:N (BV19:N)**

  modifies QEMM's QuickBoot feature so that most resident programs will no longer detect a warm boot. TSRs and device drivers sometimes monitor INT 19 (which occurs during the warm boot process) so that they can reset hardware or perform other tasks at boot time. By default, QuickBoot does not take control of the warm boot process until after the INT 19 handlers of resident software have been called. If you use BOOTVIA19:N, however, QuickBoot takes control before INT 19 is called, and programs that monitor INT 19 will not detect the warm boot. BOOTVIA19:N is more likely to cause compatibility problems than it is to eliminate them, but you can try it if you have a TSR or device driver that causes the system to fail when you do a warm boot.

- **COMPAQ386S:Y (C386S:Y)**

  tells QEMM to make the slight adjustments necessary to implement its Compaq features on the Compaq 386s. This parameter may help minimize "101 ROM error" messages that sometimes prevent warm boots on Compaq 386s systems. If you have run version 6.02 or later of Compaq's Setup program on your 386s system, you do not need the COMPAQ386S:Y parameter. You should use this parameter only with Compaq 386s systems.

- **COMPAQEGAROM:Y or N (CER:Y or N)**

  tells QEMM whether to implement the Compaq EGA ROM feature, which relocates Compaq's video ROM to increase the amount of available upper memory. Many Compaq systems keep two copies of video ROM code in upper memory: a slow ROM at C000 and a faster RAM copy of the ROM at E000. If QEMM detects this configuration on a Compaq, it makes the system use the C000 ROM and makes the E000 area available for High RAM or expanded memory mapping. In compensation, QEMM also copies the C000 ROM into faster RAM and places that RAM at C000 to avoid slowing the system.

  COMPAQEGAROM:N stops QEMM from implementing the Compaq EGA ROM feature, at a cost of as much as 32K of upper memory addresses on Compaq systems. The COMPAQFEATURES:N parameter disables this feature and two other Compaq features, and is a useful troubleshooting parameter, especially on Compaq systems. QEMM cannot be turned off if the Compaq EGA ROM feature is implemented.

- **COMPAQFEATURES:Y or N (CF:Y or N)**

  enables or disables all three of QEMM's Compaq features: Compaq EGA ROM, Compaq Half ROM, and Compaq ROM Memory. By default, QEMM enables all appropriate Compaq features on Compaq systems. You can enable/disable the three features separately with the COMPAQEGAROM, COMPAQHALFROM, and COMPAQROMMEMORY parameters. You can use COMPAQFEATURES:N for convenience, when troubleshooting.

- **COMPAQHALFROM:Y or N (CHR:Y or N)**

  tells QEMM whether to implement the Compaq Half ROM feature, which reclaims half of the Compaq system ROM's address space. Some Compaq system ROMs consist of a single 32K code area at address F000 that is duplicated at address F800. If QEMM detects this configuration on a Compaq, it makes the system use the F800-FFFF half of the System ROM, and makes F000-F7FF available for High RAM or expanded memory mapping.

  COMPAQHALFROM:N stops QEMM from implementing the Compaq Half ROM feature, at a cost of 32K of upper memory addresses on some Compaq systems. The COMPAQFEATURES:N parameter disables this feature and two other Compaq features, and is a useful troubleshooting parameter, especially on Compaq systems.

- **COMPAQROMMEMORY:Y or N (CRM:Y or N)**

  tells QEMM whether to implement the Compaq ROM Memory feature, which appropriates 128K of reserved memory that some Compaq systems use for speeding up ROMs. If QEMM detects this memory, it places it into QEMM's memory pool, thus making it available through any interface that QEMM supports (e.g., EMS, XMS, VCPI, or DPMI). In compensation, QEMM also copies the Compaq ROMs into faster RAM and places that RAM at the ROMs' addresses to avoid slowing the system.

  COMPAQROMMEMORY:N stops QEMM from implementing the Compaq ROM Memory feature. The COMPAQFEATURES:N parameter disables this feature and two other Compaq features, and is a useful troubleshooting parameter, especially on Compaq systems. QEMM cannot be turned off if the Compaq ROM Memory feature is implemented.

- **DISKBUF=nn (DB=nn)**

  tells QEMM to allocate an additional nnK of conventional memory as a disk buffer, and to send through this buffer all BIOS disk reads and writes into and out of areas where QEMM has relocated memory. The purpose of this buffer is to prevent memory addressing errors on systems that have a bus-mastering hard disk controller. The most common symptom of a conflict between QEMM and a bus-mastering hard disk controller is a system failure during the final phase of the Optimize process, or when loading any program into High RAM.

  QEMM automatically creates a 2K disk buffer whenever it detects an problem with a bus-mastering hard disk controller. A larger disk buffer uses more conventional memory but may improve disk performance; you can use the DISKBUF parameter to change the size of the disk buffer that QEMM creates by default (2K and 10K are commonly used sizes), or to force QEMM to create a disk buffer.

  You do not need the DISKBUF parameter if your bus-mastering hard disk uses the Virtual DMA Services (VDS) specification to resolve conflicts with 386 memory managers. DISKBUF will not resolve conflicts between QEMM and bus-mastering devices that are not hard disks (e.g., network adapters); such devices must support the VDS specification to work properly with 386 memory managers.

- **DISKBUFFRAME=nn (DBF=nn)**

  tells QEMM to allocate an additional nnK of conventional memory as a disk buffer, and to send through this buffer all BIOS disk accesses that transfer information into or out of the EMS page frame. QEMM should prompt you with an error message if you are using a program that requires DISKBUFFRAME. This parameter is usually needed if Stealth ROM is enabled and an EMS-using program accesses the page frame during BIOS-level disk interrupt calls. Disk caches that use EMS (e.g., PC-CACHE, Norton 5.0 N-CACHE, and the Compaq cache) and disk compression utilities (e.g., Stacker) are the programs most likely to need the DISKBUFFRAME parameter.

  For performance reasons, you may prefer to reconfigure these programs to use XMS memory instead of EMS memory if possible, thereby avoiding the need for the DISKBUFFRAME parameter. A larger disk buffer uses more conventional memory but may improve disk performance; 2K and 10K are commonly used values. DISKBUFFRAME is a weaker version of DISKBUF and is not needed when you are using DISKBUF.

- **DMA=nnn (DM=nnn)**

  specifies the size in K of QEMM's DMA (Direct Memory Access) buffer. The DMA buffer should be large enough to contain the largest DMA transfer that any hardware device in the system will make into or out of memory.

  The default values of **nnn** are 64 on PS/2 and XT-based systems, and 12 on other systems; the maximum useful value is 128. If you need a larger DMA buffer, QEMM should prompt you with an error message that tells you what size buffer to specify with the DMA parameter. The DMA buffer comes out of QEMM's memory pool and does not affect the size of conventional memory.

- **DOS4:Y (D4:Y)**

  alters the way that QEMM assigns EMS physical page numbers to physical page addresses. The DOS4:Y parameter attempts to minimize the ill effects of DOS 4's nonstandard use of expanded memory, but it does not make DOS 4's methods safe. If you are using PC- or MS-DOS 4.00 or PC-DOS 4.01, we recommend that you not use the **/X** switch to BUFFERS, VDISK, or FASTOPEN.

- **EMBMEM=nnnnn (EMB=nnnnn)**

  sets the maximum amount of extended memory that programs can allocate as EMBs (Expanded Memory Blocks) through the Extended Memory Specification (XMS). **nnnnn** is the number of K to be available as EMBs.

  By default, QEMM does not limit the amount of XMS memory that can be allocated. (The one exception is that QEMM detects when Microsoft Windows 3.0 starts up in standard mode and limits available XMS memory to 12 megabytes while it is running.) The EMBMEM parameter can be useful when any XMS-using program threatens to allocate so much memory that other programs will be unable to get memory through any specification (EMS, XMS, VCPI, or DPMI). In particular, EMBMEM is helpful when trying to run DOS programs that use EMS, VCPI, or DPMI memory inside of Microsoft Windows 3.0 standard mode.

- **EMS:N**

  tells QEMM not to provide expanded memory services, even for multitasking environments. EMS:N is sometimes useful on XT machines that have been converted to 80386 systems, where you may

want to load a different expanded memory manager to manage an EMS board and still use QEMM's other features. Otherwise, you should only use the EMS:N parameter for troubleshooting. If you want to sacrifice QEMM's expanded memory services to gain more High RAM, use the FRAME=NONE parameter instead of EMS:N.

- **EXCLUDE=xxxx-yyyy (X=xxxx-yyyy)**

  tells QEMM to make the region **xxxx-yyyy** unavailable for High RAM, expanded memory mapping or ROM mapping. For a full description, see *page 71*.

- **EXCLUDESTEALTH=xxxx (XST=xxxx)**

  tells QEMM not to use the Stealth ROM feature to relocate the ROM that begins at address **xxxx**. You should only provide the starting address of the ROM, not the entire range.

  This parameter and the EXCLUDE=xxxx-yyyy parameter are frequently used to troubleshoot problems with Stealth ROM. The EXCLUDESTEALTH parameter prevents an entire ROM from being relocated, whereas the EXCLUDE parameter can be narrowed to cover only a part of a ROM. If you find that you can use either parameter to solve your problem, you will probably be able to get more High RAM by using the EXCLUDE parameter and following the Analysis procedure (see *page 125*) to narrow the exclude as much as possible.

  If you want to use the EXCLUDESTEALTH parameter on more than one ROM region, you must use a separate EXCLUDESTEALTH parameter for each ROM. On IBM PS/2 systems, the video ROM at E000 and the system ROM at F000 are considered separate ROMs, and should be specified separately with the EXCLUDESTEALTH parameter, even though the two regions are contiguous. This parameter can, in some cases, cause a previously stable system to malfunction, especially with the Stealth frame method.

- **EXCLUDESTEALTHINT=xx (XSTI=xx)**

  tells QEMM not to intercept interrupt **xx** (where **xx** is a hexadecimal number between 0 and 7F) as part of the Stealth ROM process. Normally QEMM must intercept all interrupts that are handled by ROMs that QEMM has relocated with the Stealth ROM feature.

  The EXCLUDESTEALTHINT parameter is occasionally useful when troubleshooting problems with Stealth ROM, but it is sometimes

difficult to determine which interrupt to choose without technical insight. Furthermore, you must usually use the EXCLUDE=xxxx-yyyy parameter along with the EXCLUDESTEALTHINT=xx parameter, where xxxx-yyyy is the range of ROM addresses that contain the interrupt handler for interrupt **xx**. For information on an important use of this parameter, see **Appendix A** under the heading, "When starting up your computer you see the following message: Disabling Stealth because QEMM could not locate the ROM handler for INT XX."

- **EXTMEM=nnnnn (EXT=nnnnn)**

  specifies the amount of extended memory (in K) that should not be under QEMM's control. This parameter should only be used to provide memory for old utilities (for example, VDISK) that allocate extended memory without using the XMS, VCPI, or DPMI specifications. We recommend that, if possible, you get a new XMS-using version of your utility instead of using the EXTMEM parameter. Memory reserved by the EXTMEM parameter will be unavailable to programs that use EMS, XMS, VCPI, or DPMI.

  QEMM will not load if the EXTMEM parameter takes away so much memory from QEMM that it cannot find enough memory for its own overhead. QEMM will also refuse to load if you have a system with more than 16 megabytes of memory and EXTMEM takes away so much of the first 16 megabytes of memory that QEMM cannot use any of it. (Unmanaged memory is always left behind at the bottom of the expanded memory region.) The highest number you can safely use depends on your configuration and the amount of memory on your system, but numbers over 15,000 are likely to cause a failure no matter how much memory you have.

  You can use either the EXTMEM or MEMORY parameter to limit the memory under QEMM's control. EXTMEM=nnnnn specifies that nnnnnK should not be controlled by QEMM. MEMORY=nnnnn specifies that QEMM should control no more than nnnnnK, which much be used for its own overhead as well as its memory pool. Too large an EXTMEM number or too small a MEMORY number will cause QEMM not to load because of memory shortage.

- **FASTINT10:N (F10:N)**

  tells QEMM not to replace certain BIOS video functions with its own routines which speed up video access. The purpose of FASTINT10:N is to eliminate possible compatibility problems with video adapters

when Stealth ROM is in effect. This parameter has an effect only when used in conjunction with QEMM's Stealth ROM feature.

- **FILL:N**

  tells QEMM not to increase the size of conventional memory on systems that have unused memory addresses immediately above the top of conventional memory. By default, QEMM will fill conventional memory out to 640K on systems with less than 640K of conventional memory, and will extend conventional memory to 704K on monochrome and Hercules systems and to 736K on CGA systems. If QEMM has increased the size of conventional memory on your system, you will need the FILL:N parameter to run Microsoft Windows 3.0 or 3.1 in 386 enhanced mode, or to turn QEMM off when troubleshooting.

- **FORCEEMS:Y (FEMS:Y)**

  tells QEMM to allow programs to detect an EMS manager even when the EMS page frame is smaller than 64K due to the FRAMELENGTH=x or FRAME=NONE parameter.

  Because many programs that use expanded memory assume that the page frame is 64K in size, QEMM usually makes the standard EMS detection tests fail when you specify a number smaller than 4 to the FRAMELENGTH parameter or use the FRAME=NONE parameter. FORCEEMS:Y makes an EMS manager detectable with a small page frame or with no page frame, for the benefit of expanded memory programs that may not use all 64K of the page frame. Some VCPI programs that will not run without a page frame sometimes work if you use FORCEEMS:Y. However, some programs that use expanded memory will fail if you use FORCEEMS:Y with a small page frame or no page frame.

- **FORCESTEALTHCOPY:Y (FSTC:Y)**

  forces QEMM to copy certain tables from the video ROM, disk ROM, and system ROM into its own data areas when the STEALTHROM parameter is relocating these ROMS. These tables contain video and disk parameters, and must be accessible at all times, unlike the rest of the ROM regions, which must be accessible only at particular times.

  Normally, QEMM copies these tables to a new location in High RAM whenever Stealth ROM relocates the ROMs, unless the tables are accessible at their original locations because the EXCLUDE parameter covers those locations. FORCESTEALTHCOPY:Y makes

QEMM copy these tables to a new location even when the tables lie in regions that you have excluded.

- **FRAME=xxxx or FRAME=NONE (FR=xxxx or FR=NONE)**
  tells QEMM where to put the EMS page frame, a 64K memory region reserved for expanded memory mapping. xxxx is the four-digit hexadecimal segment address of the beginning of the page frame, which must be on a 16K boundary (i.e., the last three digits of the address should always be 000, 400, 800, or C00.)

  QEMM automatically chooses an appropriate page frame location, so normally you would not need to use the FRAME parameter. If QEMM places the page frame in an area used by another program or hardware device, it is better to use the EXCLUDE parameter to prevent access to the disputed region than it is to use the FRAME parameter to move the page frame. You may want to use the FRAME parameter to change the location of the page frame if doing so will consolidate two or more High RAM areas into one larger area. Check the appropriate Manifest screens or the QEMM.COM report display for information on your memory configuration. Be careful to avoid conflicts with memory that QEMM does not relocate, like areas that QEMM designates as Adapter RAM. Do not place the page frame at F000, even when Stealth ROM is enabled— some ROM code at the end of the F000 region must be in place at all times even with Stealth ROM in effect. QEMM will not prevent you from creating memory conflicts with a poorly chosen page frame location.

  The FRAME=xxxx parameter is most often useful when you are troubleshooting the STEALTHROM parameter. Some ROMs work with the Stealth ROM feature only when you place the page frame exactly at the beginning address of the ROM. For instance, if your VGA ROM is at addresses C000-C7FF, and you experience video problems when you use the Stealth ROM feature to relocate this ROM, you can try the FRAME=C000 parameter (assuming that the page frame at C000 does not overlay other regions filled with adapter RAM or un-Stealthed ROMs).

  When the Stealth ROM mapping method is enabled, QEMM automatically excludes the C000-C0FF region (for compatibility with Super VGA video adapters) unless the page frame is at C000. This means that placing the page frame at C000 (the default location when the Stealth ROM mapping method is enabled) gives you 4K more of upper memory to use for High RAM.

The FRAME=NONE parameter tells QEMM not to create an EMS page frame. If you use this parameter, most programs will not be able to use expanded memory. The notable exceptions are DESQview and DESQview/X, which can still multitask in expanded memory even with the FRAME=NONE parameter. In addition, you will not be able to use the STEALTHROM parameter if you use FRAME=NONE. However, if you have no programs other than DESQview and DESQview/X that use expanded memory, and if you cannot make full use of the Stealth ROM feature, you may be able to create more High RAM by using FRAME=NONE to free 64K of upper memory addresses.

- **FRAMEBUF:Y or N (FB:Y or N)**

  enables or disables QEMM's feature of breaking up disk reads into the page frame and disk writes from the page frame (when these reads and writes are done using file handles as opposed to FCBs), so that the reads and writes are diverted into DOS's buffers.

  By default, this feature is enabled when Stealth ROM is used, and disabled otherwise. FRAMEBUF:Y causes this feature to be enabled at all times; FRAMEBUF:N causes this feature to be disabled at all times. If you are not using Stealth ROM and you experience problems when running EMS-using TSRs or drivers at the same time as EMS-using applications, FRAMEBUF:Y may help.

- **FRAMELENGTH=x (FL=X)**

  tells QEMM to create an EMS page frame containing **x** 16K pages. **x** defaults to 4, which means that QEMM creates a four-page page frame covering 64K of memory addresses. Nearly all programs that use expanded memory expect a 64K page frame, so the FRAMELENGTH parameter is rarely needed. If you need to increase the amount of High RAM on your system, and if none of your expanded memory programs use all four pages of the page frame, you can use the FRAMELENGTH parameter with a number smaller than four. However, many expanded memory programs will not work properly with a small page frame, even if they do not use four pages for EMS mapping.

  You must specify the FORCEEMS:Y parameter along with the FRAMELENGTH parameter if you want an expanded memory program to use a page frame smaller than four pages.

If you specify a number greater than four to the FRAMELENGTH=x parameter, QEMM creates a page frame larger than 64K. QEMM will change the numbering of EMS physical page addresses so that the pages in the large page frame have contiguous physical page numbers. Almost no programs benefit from a page frame larger than four pages, and a large page frame decreases the area available for High RAM, so you should use this option only if you have a specific purpose. If QEMM cannot fit the large page frame into an available region in upper memory, it will put the page frame in conventional memory, decreasing the memory available to run real-mode programs.

FRAMELENGTH=0 is equivalent to the FRAME=NONE parameter—it tells QEMM not to create a page frame. See the FRAME parameter for more information.

- **GETSIZE (GS)**
  is an internal parameter set by Optimize.

- **HANDLES=nnn (HA=nnn)**
  tells QEMM how many EMS handles to provide. Each program that uses expanded memory needs at least one handle; a single program can require multiple handles if it allocates expanded memory a bit at a time. **nnn** must be in the range 16 to 255; QEMM's default is 64 EMS handles, which should be adequate for most purposes.

  If you think you need more handles, consult Manifest's Expanded Overview screen, which gives the total number of EMS handles and the number currently available. Each handle uses only 32 bytes of QEMM's memory pool.

- **HELP**
  displays a list of the QEMM386.SYS parameters and a short description of what each does. QEMM will not load if you specify this parameter.

- **HMA:N**
  tells QEMM to report the XMS High Memory Area (HMA) as already allocated, thus preventing programs from using it. This parameter is used only for troubleshooting purposes.

  The XMS:N parameter also prevents XMS-using programs from accessing the HMA or any other XMS services, but will not prevent

DESQview or DESQview/X from allocating it through a private interface to QEMM; the HMA:N parameter stops all programs, including DESQview and DESQview/X, from using the HMA, but allows programs to use other XMS services to allocate extended memory and upper memory.

- **HMAMIN=nn**
  tells QEMM not to let any program use the XMS High Memory Area (HMA) unless it requests at least nnK of the HMA's memory. **nn** must be in the range 1-63. Because only one program can allocate the HMA through QEMM, this parameter might be useful if you have more than one program asking for the HMA and you want to be sure that the most memory-efficient program gets access.

- **IBMBASIC:Y (IB:Y)**
  tells QEMM not to use the BASIC area of the IBM system ROM for High RAM or expanded memory mapping. A large area of the system ROM on IBM systems (F600-FDFF, a 32K region) contains BASIC code that is used only by some older BASIC programs. By default, QEMM makes this area available for High RAM or expanded memory mapping; use IBMBASIC:Y if you have an old BASIC program that uses the IBM BASIC area of the ROM. Newer BASIC programs like DOS 5 and 6's DOS Edit or DOS QBasic do not need the IBMBASIC:Y parameter.

- **INCLUDE=xxxx-yyyy (I=xxxx-yyyy)**
  tells QEMM to use the address range **xxxx-yyyy** for High RAM or for expanded memory mapping. For a full description, see *page 72*.

- **INCLUDE386=xxxx-yyyy (I386=xxxx-yyyy)**
  is, for QEMM's purposes, identical to the INCLUDE parameter. It is meant to be used in files (such as the MCA.ADL file) that contain parameters for QEMM's use but that may also be read by other Quarterdeck products like QEMM-50/60. Use it in these files instead of INCLUDE when an inclusion is only appropriate on an 80386 or higher processor.

- **LABEL (LB)**
  is an internal parameter set by Optimize.

- **LOCKDMA:Y (LD:Y)**

  tells QEMM not to unlock interrupts at certain times during the setup of DMA (Direct Memory Access) transfers between hardware devices and memory. By default, QEMM enables interrupts as much as possible during DMA transfers for performance reasons. Certain networks, like 10NET, will fail when QEMM is loaded unless you use the LOCKDMA:Y parameter.

- **MAPREBOOT:N (MR:N)**

  tells QEMM not to replace 4K of the System ROM with RAM. By default, QEMM shadows a particular 4K piece of the system ROM (that is, QEMM copies the ROM into RAM and makes the RAM appear in upper memory at the ROM's former location) and changes part of the ROM code so that QEMM can better detect warm reboots, which may fail if QEMM does not intervene in the reboot process. This 4K of system ROM is speeded up by being shadowed, and there is a small chance that some ROM code may fail if it executes at a faster speed.

  QEMM can often detect warm reboots even without shadowing 4K of the ROM, so you can try the MAPREBOOT:N parameter if QEMM is causing problems with your floppy drives or other devices controlled by the system ROM. This parameter returns 4K to QEMM's memory pool. When the Stealth ROM feature is enabled, QEMM does not shadow 4K of the ROM to detect reboots, and the MAPREBOOT:N parameter has no effect. If, however, the FF00-FFFF area is excluded, then QEMM shadows this 4K of the ROM even when Stealth ROM is enabled, and the MAPREBOOT:N parameter will have an effect.

- **MAPS=nnn (MA=nnn)**

  tells QEMM how many EMS alternate maps to provide. **nnn** must be in the range 0 to 255; 32 alternate maps is the default and is adequate for almost all circumstances. Alternate maps are used by multitasking environments to switch EMS contexts quickly. DESQview and DESQview/X need alternate maps to multitask high-speed communications programs effectively, and to control the video output of programs that write directly to the video hardware.

  QEMM dynamically allocates and deallocates memory out of its memory pool to satisfy requests for alternate maps; each map requested uses 4K of QEMM's memory pool. This means that there is no need to set MAPS to a lower number to save memory.

- **MEMORY=nnnnn (ME=nnnnn or MEM=nnnnn)**
  specifies the amount of extended memory (in K) that QEMM should control. The amount of memory specified by the MEMORY parameter will be used for QEMM's own overhead and for its memory pool; the remainder of the memory on the system will not be managed by QEMM. By default, QEMM manages all extended memory on the system. This parameter should only be used to provide memory for old utilities (for example, VDISK) that allocate extended memory without using the XMS, VCPI, or DPMI specifications. We recommend that, if possible, you get a new XMS-using version of your utility instead of using the MEMORY parameter. Memory reserved by the MEMORY parameter will be unavailable to programs that use EMS, XMS, VCPI, or DPMI.

  QEMM will not load if the MEMORY parameter restricts QEMM so severely that it cannot find enough memory for its own overhead. QEMM will also refuse to load if you have a system with more than 16 megabytes of memory and MEMORY restricts QEMM so that it cannot use any of the first 16 megabytes. (Unmanaged memory is always left behind at the bottom of the expanded memory region.) The lowest number you can safely use depends on your configuration, but leaving over 15,000K of unmanaged memory is likely to cause a failure. See the EXTMEM parameter.

- **OFF (OF)**
  See **AUTO ON OFF**.

- **OLDDV:Y (ODV:Y)**
  makes changes to QEMM so that it is compatible with DESQview versions 1.3 and 2.00. You do not need this parameter if you use DESQview 2.01 or later.

- **ON**
  See **AUTO ON OFF.**

- **PAUSE**
  tells QEMM to pause for input when it loads, regardless of whether it detects an error. You can then hit the Esc key to unload QEMM or any other key to continue the loading process. The PAUSE parameter gives you an easy way to prevent QEMM from loading when you are trying out new parameters that may make the system unstable.

- **PAUSEONERROR:N (PE:N)**
  tells QEMM not to pause to give an error message. QEMM normally displays an error message and pauses for input when you specify an incorrect parameter, when it detects a microchannel adapter not listed in the MCA.ADL file, and in other circumstances. When you use this parameter, QEMM will display error messages but will not pause. You can use PAUSEONERROR:N if QEMM is giving you an error message about a harmless condition that you prefer not to address (e.g., an "Unknown MCA Adapter ID" error for a microchannel adapter that you are certain uses no addresses in upper memory).

- **PENTIUM:VME:N (P:VME:N)**
  tells QEMM not to take advantage of the Virtual Mode Extensions features, which let QEMM optimize the performance of software interrupts on Pentiums and some 80486 systems. QEMM's default on these systems is to pass many software interrupts directly to real-mode programs, thus saving the overhead that would be required to pass these interrupts from protected-mode handlers to real-mode handlers. This practice ordinarily causes no problems unless a program changes QEMM's internal tables on the assumption that the program will then be able to intercept software interrupts in protected mode. In this unusual circumstance, PENTIUM:VME:N will let such a program work, at the cost of some system performance.

- **PENTIUM:PSE:N (P:PSE:N)**
  tells QEMM not to take advantage of the Page Size Extension feature, which lets QEMM reduce extended memory overhead on Pentiums and some 80486 systems. QEMM's default on these systems is to create a page directory in such a way that many second-level page tables are not required, thus saving 4K of extended memory for every 4 megabytes of memory on the system. This practice ordinarily causes no problems unless a program accesses QEMM's page directory and does not understand these new capabilities on Pentium and late 80486 systems. In this unusual circumstance, PENTIUM:PSE:N will let such a program work, at the cost of greater overhead for QEMM in extended memory.

- **RAM or RAM=xxxx-yyyy**
  specifies that QEMM should create High RAM in upper memory. For a full description, see *page 72*.

- **REGION:n (R:n)**

  tells QEMM in which region of High RAM to place pieces of its own code, as well as 12-13K of video and disk parameter tables that it relocates when the Stealth ROM feature is enabled. The Optimize program will place the REGION parameter on the QEMM386.SYS device driver line when it is appropriate; you should normally not add this parameter manually.

- **RESPONSEFILE[:file] (RF[:file])**

  tells QEMM to look into a file to determine which High RAM region to use to load parts of itself into upper memory. RESPONSEFILE may be followed by a colon and a filename (or pathname, if necessary). If you do not specify a file, QEMM will look for the environment variable LOADHIDATA to determine the name and location of the file; the default is \QEMM\LOADHI.RF. The LOADHIDATA environment variable is created when you run Optimize on a system that has an MS-DOS 6 multiple configuration or when you run OPTIMIZE with the RESPONSE parameter.

  If you are using MS-DOS 6 multiple configurations, Optimize will place the RF parameter instead of the REGION (R:n) parameter on the QEMM386.SYS device line. The response file itself will contain the REGION parameter (R:n), specifying which High RAM region to use. For information on the REGION parameter, see the description of LOADHI's REGION parameter in **Chapter 8**.

- **ROM, ROM=xxxx or ROM=xxxx-yyyy**

  tells QEMM to speed up the operation of ROMs by copying them into faster RAM and making the RAM appear in upper memory in place of the ROMs. For a full description, see ***page 73***.

- **ROMHOLES:N (RH:N)**

  tells QEMM not to detect unused areas of ROM and make their memory addresses available for High RAM or expanded memory mapping. By default, QEMM examines the system ROM for areas that are either empty or used only during the system's power-on self test, and marks these areas as available for other uses. This feature is implemented only when QEMM does not relocate the entire system ROM with the Stealth ROM feature.

  If QEMM mistakenly reclaims a ROM area that is in use, you may experience problems with your floppy drives or with other devices or

functions controlled by ROMs. Both the ROMHOLES:N and the EXCLUDE=xxxx-yyyy parameter prevent QEMM from finding "ROM holes," and the address range of the EXCLUDE parameter can be adjusted to leave you some of the benefits of QEMM's reclamation of ROM addresses; therefore, if the ROMHOLES:N parameter fixes a problem, you should probably use QEMM's Analysis procedure (see "The Analysis Report" on *page 125*) to find appropriate EXCLUDE parameters to use instead of ROMHOLES:N.

- **SHADOWRAM:LEAP, NEAT, NEC, OPTI, PEAK, SCAT, TOPCAT, 386, or NONE (SH)**

  forces QEMM to reclaim various types of "shadow memory" for its memory pool, or tells QEMM not to reclaim it. Shadow memory is 384K of RAM that can be made to appear in the upper memory area on some systems. It serves the same function as QEMM's ROM parameter: systems with shadow memory can copy ROMs and other memory in the upper memory area into this 384K of RAM to speed up system activities.

  By default, QEMM automatically detects various kinds of shadow memory and reclaims any unused portions for its memory pool. In the unlikely event that QEMM fails to detect a type of shadow memory that it knows how to manage, you can use the SHADOWRAM parameter to force QEMM to recognize the shadow memory. **LEAP**, **NEAT**, **PEAK**, **SCAT**, and **386** refer to different varieties of Chips & Technologies ShadowRAM; **NEC** refers to a variant of ShadowRAM that is found on some NEC systems; and **OPTI** and **TOPCAT** refer to the shadow memory of the OPTI and VLSI TOPCAT chip sets, respectively.

  The SHADOWRAM:NONE parameter disables this QEMM feature, and is a common troubleshooting option. You should try this parameter if your system fails at the point when QEMM loads, or if your system reboots when QEMM loads.

  You must sometimes enable shadow memory in your system setup for QEMM to detect it. If your system lets you reconfigure shadow memory as extended memory, you should do so and let QEMM shadow your ROMs with its ROM parameter. This typically increases QEMM's memory pool by 96K (192K if you do not use the ROM parameter) on EGA/VGA systems.

- **SORT:Y**

  tells QEMM to test the speed of all system memory and to use the fastest memory first. This parameter can sometimes speed up systems that contain memory chips of different speeds—especially systems with slower motherboard memory and faster memory on an add-in card. However, Microsoft Windows 3.0 and 3.1 will not run in 386 enhanced mode if the SORT:Y parameter makes QEMM use faster extended memory in place of conventional memory. You can use Manifest's Expanded Timings screen to see if different parts of your memory run at different speeds.

- **STEALTHROM:x (ST:x)**

  enables the Stealth ROM feature—the QEMM technology that relocates ROMs from upper memory so that their addresses can be used for High RAM or for expanded memory mapping. For a full description, see *page 74*.

- **SUSPENDRESUME (SUS) or SUSPENDRESUME:nn (SUS:nn)**

  forces QEMM to enable its support for Suspend/Resume, a feature built into many laptop systems that allows you to operate the laptop on low power when it is not in use and to restore the system to its previous state when you return to it.

  QEMM detects Suspend/Resume automatically on some systems and will enable Suspend/Resume support on them without the SUSPENDRESUME parameter. If your laptop supports the Suspend/Resume feature and that feature does not work properly after installing QEMM, the SUSPENDRESUME parameter will make QEMM search for the hardware interrupt that the feature uses, and enable its Suspend/Resume support if it finds the interrupt.

  If QEMM cannot find the Suspend/Resume hardware interrupt, you can use the SUSPENDRESUME:nn parameter, where **nn** is the hexadecimal number of the appropriate hardware interrupt. The documentation for your laptop may tell you which interrupt the Suspend/Resume feature uses; if it does not, 2, D, 72, 73, and 77 are the most likely possibilities.

- **TASKS=nn (TA=nn)**

  tells QEMM how many interrupts it can keep track of at one time when it is in protected mode. **nn** must be in the range 2 to 40; the default is 16. Though you will probably not need to use the

TASKS=nn parameter, you can try increasing the value of nn if your system halts suddenly in the middle of high-speed communications or other interrupt-intensive activity.

- **TOKENRING:N (TR:N)**
  tells QEMM not to detect the presence of a Token-Ring adapter. By default, QEMM will use special methods to ensure that it does not place High RAM or allow expanded memory mapping in upper memory areas occupied by the Token-Ring card's adapter RAM and ROM. You should try the TOKENRING:N parameter if you have another adapter that does not work or cannot be detected when QEMM is loaded. If this parameter solves your problem, and if you have a Token-Ring adapter as well, you can simply use the EXCLUDE=xxxx-yyyy parameter for any Token-Ring memory addresses that QEMM has not already detected.

- **TOPMEMORY:N (TM:N)**
  tells QEMM not to try to reclaim "top memory," memory located just below the 16-megabyte point on some systems and reserved for speeding up ROMs and for other system functions. By default, QEMM automatically detects unused portions of top memory and places them into QEMM's memory pool, thus making it available through any interface that QEMM supports. TOPMEMORY:N disables this QEMM feature, and is a common troubleshooting option. You should try this parameter if your system fails at the point when QEMM loads, or if your system reboots when QEMM loads.

- **TRAP8042:Y (T8:Y)**
  tells QEMM to monitor output to the 8042 keyboard controller. The keyboard controller contains a bit that enables A20, a hardware address line that determines whether programs can access extended memory. You can use the TRAP8042:Y parameter if you believe that a program that is accessing extended memory is interfering with QEMM's memory management. You may also need TRAP8042:Y to warm reboot successfully on Intel Inboard-AT systems.

  Forcing QEMM to monitor the controller port adds overhead to the processing of keystrokes. If you are using a resident program or device driver that reads keystrokes incorrectly from the keyboard controller port, TRAP8042:Y can sometimes cause the shift states on your keyboard to become reversed, the arrow keys on your cursor keypad to produce numbers instead of cursor movement, and other

problems with the keyboard or with the mouse on a PS/2-style mouse port.

- **UNMAPFREEPAGES:Y or N (UFP:Y or N)**
  tells QEMM whether to remove pages of expanded memory from the EMS page frame when an EMS-using program has freed the expanded memory handle associated with these pages.

  UNMAPFREEPAGES:Y is the default when Stealth ROM is enabled—this makes it possible to use Stealth ROM along with certain Super VGA video cards without excluding parts of the video ROM area. However, UNMAPFREEPAGES:Y is incompatible with certain EMS-using applications, including some versions of Glyphix, VP Planner, and 1DIR Plus. If you have a problem running an EMS-using program when Stealth ROM is enabled, you should try UNMAPFREEPAGES:N, though this parameter may force you to exclude parts of the C000-C7FF area. UNMAPFREEPAGES:N is QEMM's default when Stealth ROM is not in effect, and there is usually no need to try UNMAPFREEPAGES:Y when Stealth ROM is not active.

- **UNUSUALEXT:Y (UX:Y)**
  tells QEMM to change its method of determining the amount of extended memory on the system. You will probably not need this parameter, but you can try it if your system hangs as soon as QEMM loads. UNUSUALEXT:Y may cause QEMM.COM's Memory report and Manifest's QEMM Memory screen to have inaccurate extended memory reports.

- **USERAM=xxxx-yyyy (UR=xxxx-yyyy)**
  tells QEMM that the system has placed RAM in the address range **xxxx-yyyy** that QEMM should reclaim and place in its memory pool. If your system is using a type of shadow memory that QEMM cannot detect automatically, you can specify the USERAM parameter for any regions of upper memory that do not contain ROMs, video RAM, or adapter RAM. QEMM will add the shadow memory in these unused regions to its memory pool, and automatically make these regions available for High RAM or expanded memory mapping.

  Some machines with more than 16 megabytes of memory do not report the memory above 16 megabytes in a way that QEMM can detect it. You can specify the USERAM parameter to tell QEMM about RAM at extended memory addresses that QEMM cannot

detect automatically. For information on finding out if QEMM is detecting memory above 16 megabytes and what addresses to use with USERAM, see the SCANMEM program in **Chapter 12**.

- **USEXMS:N**

  tells QEMM not to allocate memory from a previously loaded XMS manager, if one exists, and to obtain extended memory only through the BIOS interface. Normally, QEMM will allocate all memory away from a previously loaded XMS manager (e.g., DOS's HIMEM.SYS). USEXMS:N also prevents QEMM's default practice of letting a previously loaded XMS manager do all manipulations of the A20 address line, which determines whether programs can access extended memory. This parameter is rarely used.

- **VCPISHARE:Y (VS:Y)**

  tells QEMM to make updates to VCPI protected-mode programs' first page table, which contains the VCPI program's map of the first four megabytes of memory. This parameter may help some VCPI programs run properly, especially when the VCPI program uses EMS memory or when it is being swapped out of memory by DESQview. However, VCPISHARE:Y can also cause some VCPI programs to fail.

- **VDS:N**

  disables QEMM's support of the Virtual DMA Services (VDS) specification, which resolves incompatibilities between 386 memory managers and bus-mastering hardware devices. This parameter should be used only for troubleshooting purposes.

- **VIDEOFILL:N (VF:N)**

  tells QEMM not to increase the size of conventional memory on systems that do not use the video areas just above 640K. By default, QEMM will extend conventional memory to 704K on monochrome and Hercules systems and to 736K on CGA systems, unless some hardware or software on your system has placed its code or data just below 640K and QEMM cannot move that code or data. If QEMM has increased the size of conventional memory on your system, you will need the VIDEOFILL:N or the FILL:N parameter to run Microsoft Windows 3.0 or 3.1 in 386 enhanced mode, or to turn QEMM off when troubleshooting. The VIDEOFILL:N parameter is a weaker version of the FILL:N parameter, which prevents filling conventional memory into areas both above and below 640K.

- **VIDEORAM:N (VR:N)**
  tells QEMM not to create High RAM in upper memory areas reserved for video RAM (640K-768K). By default, QEMM will create High RAM in all available areas of upper memory when the RAM parameter is in effect, including any unused areas between 640K and 768K.

  You can use either the VIDEORAM:N parameter or the VIDRAMEGA parameter if you plan to use the VIDRAM program's VIDRAM ON or VIDRAM ON EGA option to extend conventional memory on a color EGA or VGA system, and you want conventional memory to be extended to 736K instead of 704K. (The larger amount of conventional memory comes at the expense of 32K of High RAM at B000-B7FF.) Of the two parameters, VIDRAMEGA may be the safer option if you have programs that make extensive use of expanded memory mapping; VIDEORAM:N, on the other hand, has the advantage of leaving the B000-B7FF area available for expanded memory mapping by DESQview or DESQview/X.

  If you want to stop QEMM from putting High RAM in the video RAM areas for troubleshooting purposes, you should normally use the EXCLUDE=A000-BFFF parameter instead of the VIDEORAM:N parameter. See **Chapter 6** for information on the VIDRAM Program.

- **VIDRAMEGA (VREGA)**
  tells QEMM to make certain video regions unavailable for High RAM or expanded memory mapping. It is equivalent to the parameter EXCLUDE=A000-B7FF.

  The VIDRAMEGA parameter is useful when you are using QEMM's VIDRAM program. If you use VIDRAM ON or VIDRAM ON EGA, the VIDRAMEGA parameter will insure that VIDRAM increases the size of conventional memory by 96K instead of 64K. Also, VIDRAMEGA is useful if you are using VIDRAM to increase the memory available to a DOS window inside Microsoft Windows' 386 enhanced mode, and you want the Windows DOS window to increase in size by 96K instead of 64K. In both cases, VIDRAMEGA will decrease by 32K the amount of upper memory available for High RAM and expanded memory mapping on most EGA/VGA systems. See **Chapter 6** for information on VIDRAM.

- **VIDRAMEMS (VREMS)**
  tells QEMM to make the video regions just above 640K available for expanded memory mapping, but not to fill them with High RAM or to extend conventional memory into them.

  If you are using the VIDRAM program to increase the sizes of all your windows in DESQview (or DESQview/X running in 8514 graphics mode), you will need the VIDRAMEMS parameter. It will decrease by 32K the amount of upper memory available for High RAM on most EGA/VGA systems. This parameter may cause problems with EMS-using programs that make extensive use of expanded memory mapping. Do not use VIDRAMEMS with versions of DESQview prior to Version 2.26. See **Chapter 6** for information on VIDRAM.

- **VIRTUALHDIRQ:N (VHI:N)**
  tells QEMM not to disable the "advanced disk features" of some disk caches when Stealth ROM is enabled. Sophisticated disk caches often take advantage of a BIOS feature that lets them give processor time to applications during the waiting time while the disk controller is writing a sector to disk. Unless the disk cache takes the proper precautions, these advanced disk features are incompatible with QEMM's Stealth ROM feature. Therefore, QEMM disables advanced disk features by default whenever Stealth ROM causes QEMM to intercept the hard disk's software interrupt, INT 13.
  VIRTUALHDIRQ:N tells QEMM to permit advanced disk features even when Stealth ROM causes QEMM to intercept INT 13.

  When Stealth ROM is not enabled or is not causing INT 13 to be intercepted, the VIRTUALHDIRQ parameter has no effect.

  Disk caches that have advanced disk features that are compatible with Stealth ROM will communicate with QEMM to enable these features; it is therefore unwise to use the VIRTUALHDIRQ:N parameter when Stealth ROM is enabled and you are loading a disk cache.

- **VXDDIR=directory_name**
  tells QEMM the location of its .VXD files, which it uses to provide various features when Microsoft Windows 3.0 and 3.1 is running in 386 enhanced mode. You need this parameter if you use a diskless workstation on a network or if your .VXD files are not in the directory from which QEMM is loaded.

- **WATCHDOG=0,1, or 2 (WD=0, 1 or 2)**
  tells QEMM the type of hardware, if any, that is available on your system for QEMM to use to implement Level 1 of DESQview and DESQview/X's Protection Level feature. Protection Level 1 notifies the user when a program has locked interrupts for too long—a common cause of system failures. Under normal circumstances, QEMM supports this DESQview and DESQview/X feature only on IBM PS/2 systems and on Compaq systems, each of which contains watchdog timer hardware that QEMM automatically detects.

  You can use the WATCHDOG=1 or WATCHDOG=2 parameter if your system contains watchdog timer hardware that QEMM knows how to use but does not detect. WATCHDOG=1 tells QEMM to look for a PS/2-style watchdog timer, and WATCHDOG=2, a Compaq-style or an EISA-style watchdog timer. You should note in particular that QEMM does not automatically detect the watchdog hardware on EISA systems, which need WATCHDOG=2 to make Protection Level 1 work at all in DESQview and DESQview/X. WATCHDOG=0 is the default on systems that are not PS/2s or Compaqs, and disables Protection Level 1 in DESQview and DESQview/X. If you experience problems when using Protection Level 1 in DESQview or DESQview/X, you can try the WATCHDOG=0 parameter.

- **WINDOWS3:N (W3:N)**
  disables QEMM's special support for Microsoft Windows 3.0 in standard and 386 enhanced modes and Microsoft Windows 3.1 in 386 enhanced mode. When you specify the WINDOWS3:N parameter, you can run Microsoft Windows 3.0 only in real mode, and Microsoft Windows 3.1 only in standard mode. This parameter changes slightly the information that QEMM returns to some EMS calls, and therefore has a small chance of preventing problems with programs that use expanded memory.

- **WINSHRINKUMBS:N (WSU:N)**
  tells QEMM not to shrink the amount of available High RAM when Microsoft Windows 3.0 or 3.1 starts up in 386 enhanced mode. By default, QEMM gets rid of most unused areas of High RAM when it sees Windows start up in this mode, because freeing these areas of upper memory usually increases the size of DOS windows by 8-24K. WINSHRINKUMBS:N makes it possible for programs running inside Microsoft Windows' 386 enhanced mode to use available High

RAM, but also will frequently decrease the size of DOS windows when Windows is running in 386 enhanced mode.

- **XBDA:N, L, H, or F**

  tells QEMM how to treat the Extended BIOS Data Area (XBDA). The XBDA (Extended BIOS Data Area) is a RAM region on IBM PS/2s and some PC clones that is reserved to contain hardware information beyond that contained in the BIOS data area. Usually, the XBDA is located at the top of conventional memory, and can be an obstacle to effective memory management.

  By default, QEMM detects the XBDA on systems that have it, and moves it to High RAM. However, if QEMM detects that your PC has a Suspend/Resume feature, or if you place the SUSPENDRESUME parameter on the QEMM386.SYS line, QEMM will by default move the XBDA to a low address in conventional memory (in case the BIOS accesses it during a suspend process). In addition, some IBM ThinkPad notebook PCs cause QEMM to move the XBDA to low conventional memory to avoid compatibility problems. QEMM will move the XBDA to low conventional memory if there is no available High RAM.

  On monochrome, Hercules and CGA systems, moving the XBDA to High RAM or to low conventional memory lets QEMM perform video filling. On EGA/VGA systems, moving the XBDA makes it possible to use VIDRAM and also increases the size of most DESQview and DESQview/X windows by 16K.

  XBDA:H (the default) forces QEMM to put the XBDA in High RAM if there is any available. This parameter can save you 1K of conventional memory, but if you have a notebook PC with a Suspend/Resume feature, that feature may not work properly. Some IBM ThinkPad models will also malfunction if you force the XBDA into High RAM.

  XBDA:N tells QEMM not to move the XBDA. XBDA:N is a common troubleshooting option, and is needed whenever a ROM or an application assumes that the XBDA is located at the top of conventional memory. A symptom of this problem is usually a system crash at boot time or later, or failure to access a disk.

  XBDA:L forces QEMM to move the XBDA to low conventional memory instead of High RAM; this has the same benefits as XBDA:H except that it does not give you the additional 1K more of

conventional memory. This parameter is therefore a less drastic solution to XBDA-related problems than XBDA:N, although XBDA:N will solve some problems that XBDA:L will not.

XBDA:F tells QEMM to try to relocate any piece of data or code at the top of conventional memory, even if it is not an Extended BIOS Data Area. XBDA:F can also move multiple data or code areas from the top of conventional memory. By default, QEMM will not move such unknown objects because of possible compatibility problems. When you specify the XBDA:F parameter, QEMM tries to move the XBDA-like object into High RAM. However, if the system does not function properly with the XBDA-like object in High RAM, you can also try to move it into low conventional memory by using both the F behavior and the L behavior (i.e., XBDA:FL).

- **XMS:N**
  tells QEMM not to provide support for the Extended Memory Specification (XMS), which many programs use to access extended memory and upper memory. You should normally only use this parameter for troubleshooting. The XMS:N parameter will not prevent DESQview and DESQview/X from using the High Memory Area (HMA); use the HMA:N parameter for that purpose.

Using the ROM and STEALTHROM Parameters Together

When you specify the STEALTHROM:x parameter to QEMM, ROM code is managed by QEMM in such a way that upper memory ROM areas are then free for High RAM or expanded memory mapping.

QEMM processes its parameters sequentially—if two mutually exclusive parameters appear on the QEMM386.SYS line in the CONFIG.SYS file, the one that comes later overrides the earlier one. This is primarily important when two QEMM parameters make claims on the same address range.

One situation in which the ordering of parameters is important is when the ROM parameter and the STEALTHROM:x parameter are both specified. There is no conflict or overlap between the function of these two parameters—the presence of the ROM parameter simply means that the STEALTHROM:x parameter will be relocating fast ROMs instead of slow ones. However, when you specify another parameter that affects a ROM address range, whether that parameter affects the mapping of ROM into faster RAM depends on if it

precedes or follows the STEALTHROM:x parameter. Here is an example of such a parameter set:

ROM EXCLUDE=FC00-FFFF STEALTHROM:M

Here, the ROM parameter tells QEMM to map ROM areas into faster RAM. The EXCLUDE=FC00-FFFF parameter prevents RAM from being mapped into the FC00-FFFF area, so the ROMS in that area will not be mapped into RAM. (The ROM code that controls the floppy drive may be in the FC00-FFFF range, and in some cases, floppy drive access is impaired if the ROM code that controls the floppy drive controller is speeded up.)

The following parameter set will have a different effect:

ROM STEALTHROM:M EXCLUDE=FC00-FFFF

In this case, the EXCLUDE=FC00-FFFF parameter no longer affects the mapping of ROM into faster RAM; the sequencing of the parameters ensures that the ROM has been relocated by the STEALTHROM:M parameter before the exclusion comes into play. The EXCLUDE=FC00-FFFF parameter still tells QEMM not to make that range available for High RAM or EMS mapping. The ROM code at FC00-FFFF can still be accessed at its old address. By placing the EXCLUDE after the STEALTHROM:M parameter, you can speed up the entire system ROM and still prevent other access to the excluded area.

In addition to EXCLUDE, there are other parameters that can affect the ROM parameter's range, and will therefore behave differently on different sides of the STEALTHROM:x parameter. They are:

INCLUDE=xxxx-yyyy
INCLUDE386=xxxx-yyyy
ADAPTERROM=xxxx-yyyy
ADAPTERRAM=xxxx-yyyy
RAM=xxxx-yyyy

When you specify the RAM parameter without any address range, it does not require the same care as RAM=xxxx-yyyy when you are ordering your parameters. RAM without any addresses simply means "Put High RAM everywhere that it will not conflict with anything else," and will not change the extent of the ROM parameter's influence.

If you specify an address range with the ROM parameter (e.g., ROM=C000-C7FF), you must place the ROM parameter before the STEALTHROM:x parameter; otherwise the ROM parameter will be ignored.

Using Parameters from Previous Versions of QEMM

If you have upgraded from an QEMM version 6 or earlier, be aware that many of the old parameters have new names. You can still use the older parameter names if you like. This section provides you with a list of the old parameter names cross referenced with the parameters' new names. Some parameters have abbreviations which are listed in parentheses.

Old Name	New Name
COMPAQ386S (C386S)	COMPAQ386S:Y (C386S)
COMPAQEGAROM (CER)	COMPAQEGAROM:Y (CER)
COMPAQHALFROM (CHR)	COMPAQHALFROM:Y (CHR)
COMPAQROMMEMORY (CRM)	COMPAQROMMEMORY:Y (CRM)
DONTUSEXMS (DUX)	USEXMS:N
DOS4 (D4)	DOS4:Y (D4)
FORCEEMS (FEMS)	FORCEEMS:Y (FEMS)
FORCESTEALTHCOPY (FSTC)	FORCESTEALTHCOPY:Y (FSTC)
IGNOREA20 (IA)	TRAP8042:Y (T8) (the default is reversed from earlier versions)
LOCKDMA (LD)	LOCKDMA:Y (LD)
NOCOMPAQFEATURES (NCF)	COMPAQFEATURES:N (CF)
NOEMS	EMS:N
NOFILL (NO)	FILL:N
NOHMA	HMA:N
NOPAUSEONERROR (NOPE)	PAUSEONERROR:Y (PE)
NOROM (NR)	MAPREBOOT:N (MR)
NOROMHOLES (NRH)	ROMHOLES:N (RH)
NOSHADOWRAM (NOSH)	SHADOWRAM:NONE (SH)
NOTOKENRING (NTR)	TOKENRING:N (TR)
NOTOPMEMORY (NT)	TOPMEMORY:N (TM)

Old Name	New Name
NOVDS	VDS:N
NOVIDEOFILL (NV)	VIDEOFILL:N (VF)
NOVIDEORAM (NVR)	VIDEORAM:N (VR)
NOWINDOWS3 (NW3)	WINDOWS3:N (W3)
NOXBDA (NX)	XBDA:N
NOXMS	XMS:N
OLDDV (ODV)	OLDDV:Y (ODV)
UNUSUALEXT (UX)	UNUSUALEXT:Y (UX)

NOTES:

8 The LOADHI Programs

This chapter is a technical reference for QEMM's LOADHI programs which load device drivers and TSRs into High RAM. QEMM's Optimize program automatically adds LOADHI commands where necessary to cause these items to be loaded high. Read this chapter if you want to find out what items are loaded high or learn how to use the LOADHI commands.

QEMM's LOADHI.SYS and LOADHI.COM programs load TSRs and device drivers into available regions of High RAM. The Optimize program adds the necessary LOADHI statements to your CONFIG.SYS and AUTOEXEC.BAT files to load these items high. The purpose of loading TSRs and device drivers high is to free up more conventional memory for running DOS programs. If LOADHI cannot find enough High RAM to load an item high, it will use conventional memory instead.

To load items high, your PC must have High RAM. The installation automatically creates High RAM by adding the RAM parameter to the QEMM386.SYS line of your CONFIG.SYS file. If you do not have High RAM, you can run Optimize to create it and to add LOADHI statements to your CONFIG.SYS and AUTOEXEC.BAT files (see **Chapter 3** for information on Optimize).

This chapter explains when and how you can use the LOADHI programs. There are several optional parameters that determine LOADHI's effectiveness in freeing up conventional memory. Freeing up this memory may enable you to:

- Run programs that would not fit in memory before.
- Add TSRs you have been doing without.
- Speed up memory-starved programs that no longer need to go to disk for their data.
- Increase the memory available to applications running in multitasking programs such as Microsoft Windows, DESQview, and DESQview/X.

If LOADHI is unable to load a particular item high, the item will load into conventional memory. Thus, your device drivers and TSRs will be available even if they are not loaded high.

Using LOADHI

Both LOADHI.SYS and LOADHI.COM load programs into High RAM. You use LOADHI.SYS to load device drivers with an appropriate **DEVICE=** statement in your CONFIG.SYS file. You use LOADHI.COM to load programs from the DOS prompt, or from within your AUTOEXEC.BAT or other batch file. Both LOADHI programs support the same set of parameters which you can use to modify the way LOADHI normally allocates and uses High RAM. These parameters are described later in this chapter.

The LOADHI Report

The LOADHI report tells you which High RAM addresses are in use, the amount of High RAM used and what resources are using it. This report is your starting point if you need to use LOADHI's optional parameters.

To display the LOADHI report:

```
C:\>loadhi
```

Region	Area	Size	Status
1	B001 - B07A	1.8K	Used (QDPMI)
1	B07B - B0AB	0.7K	Used (SETVER)
1	B0AC - B0B7	0.1K	Used (DV)
1	B0B8 - B5D7	20K	Used (LSL)
1	B5D8 - B7FE	8.5K	Used (DV)
2	D000 - D38B	14K	Used (QEMM386)
2	D38C - D391	0.1K	Used (DV)
2	D392 - DA9A	28K	Used (SMARTDRV)
2	DA9B - DAA4	0.1K	Used (DV)
2	DAA5 - DED0	16K	Used (MOUSE)
2	DED1 - DEDC	0.1K	Used (DV)
2	DEDD - DFDE	4K	Used (NE2000)
2	DFDF - E3C0	15K	Used (IPXODI)
2	E3C1 - E8EB	20K	Used (TCPIP)
2	E8EC - F364	41K	Used (NETX)
2	F365 - FFA5	49K	Used (DV)

The LOADHI report

- **At the DOS prompt, type LOADHI and press Enter ↵.**

High RAM areas may be scattered throughout the upper memory area due to the presence of ROM and adapter RAM. Each contiguous area of memory converted into High RAM is called a *region* and QEMM gives each region a number (listed in the left-most column of the report). These regions may vary in size.

To load items high, LOADHI allocates blocks of memory from these regions. In the LOADHI report, the Area column shows each block of memory that has been allocated. Each block allocated reduces the amount of available memory in the region. Any unused areas will be marked as Available.

If you want to see the number of regions, their address ranges, and sizes, display the LOADHI report before loading anything high. The sum of the regions' sizes gives you the maximum amount of space available to load items high.

Loading Device Drivers High With LOADHI.SYS

You use LOADHI.SYS in your CONFIG.SYS file to load device drivers into High RAM at system startup time. LOADHI.SYS can load a device driver file that contains any number of device driver definitions.

If you have not already run QEMM's Optimize program, we suggest you do so. Optimize adds the appropriate LOADHI statements to your CONFIG.SYS file to load your device drivers high. This section is for those who want to manually add or edit LOADHI statements in CONFIG.SYS.

Any statement in your CONFIG.SYS file that begins with the keyword **DEVICE=** (or DEVICEHIGH=) tells DOS to load the device driver following that keyword. Some common devices are memory managers, add-on peripheral device drivers (e.g., disk drive, mouse), and extensions to existing devices (e.g., DOS's ANSI.SYS driver).

A few device drivers are location-sensitive and will not work properly when loaded high and some other device drivers may require that you experiment with LOADHI parameters (you should not load QEMM386.SYS, DOSDATA.SYS or DOS-UP.SYS high). But generally, you should have no trouble loading device drivers high. To load a device driver high:

- **Add DEVICE=C:\QEMM\LOADHI.SYS, followed by the device driver's full pathname and any parameter the device driver uses, to your CONFIG.SYS file.**

  All LOADHI statements must appear after the QEMM386.SYS line. Be sure to include any parameters you normally use with your device driver.

The ANSI.SYS device driver is included with DOS, so we will use it to demonstrate how to load a device high. Normally, you load ANSI.SYS with the line DEVICE=C:\DOS\ANSI.SYS. To load it high, you would use

DEVICE=C:\QEMM\LOADHI.SYS C:\DOS\ANSI.SYS

If you need control over the placement of device drivers or if you have problems loading a device driver high, see "The LOADHI Parameters" later in this chapter.

Loading TSRs High With LOADHI.COM

You can use LOADHI.COM to load TSRs into High RAM. You can load TSRs from your AUTOEXEC.BAT file, another batch file or from the DOS prompt.

If you have not already run QEMM's Optimize program, we suggest you do so. Optimize adds the appropriate LOADHI statements to TSR lines in your AUTOEXEC.BAT file (and any batch files called by AUTOEXEC.BAT) to load those TSRs high. This section is for those who want to manually add or edit LOADHI statements in AUTOEXEC.BAT or load a TSR high by typing a command at the DOS prompt.

To load a TSR high:

- **Type C:\QEMM\LOADHI followed by the command you normally use to load the TSR. You could add this statement to AUTOEXEC.BAT or type it at the DOS prompt.**

To demonstrate how to load a TSR high, we will use an imaginary TSR program called DOITALL. To run DOITALL, you normally use the command C:\UTILS\DOITALL /R. This statement could be in your AUTOEXEC.BAT file, in some other batch file, or you could simply type it at the DOS prompt.

To load DOITALL into High RAM, change the above statement to C:\QEMM\LOADHI C:\UTILS\DOITALL /R

If you need control over the placement of TSRs or if you have problems loading a TSR high, see "The LOADHI Parameters" later in this chapter.

The LOADHI Parameters

LOADHI has several optional parameters that may help if you have trouble loading one or more of your programs into High RAM.

The LOADHI command has the following syntax:

LOADHI [loadhi-parameters] program [program-parameters]

The brackets in the statement indicate that the item specified is optional. The **program** represents the device driver or TSR you are loading.

Note the following:

- All LOADHI options begin with a forward slash (/).
- Following the slash is the option name, or its abbreviation.
- If the option can take a value, then the next character must be a colon (:) followed by the value.

For example, **/LARGEST[:n] (/L)** means you can use any of the following:

/LARGEST
/L
/LARGEST:2
/L:2

The LOADHI parameters are described below. You can combine options if necessary. Some parameters have an abbreviation you can use instead of the full name; abbreviations are listed in parentheses below. If a parameter has an optional value, it is enclosed in square brackets ([ ]).

- **/BESTFIT (/B)**
  tells LOADHI to use the smallest block of memory in which the program fits. Using BESTFIT tends to leave larger regions for loading larger items. See also **SIZE** and **GETSIZE**.

- **/ENVHI (/EH)**
  tells LOADHI.COM to load the DOS environment into upper memory. Because some versions of the Novell NetWare shell can fail if the environment is loaded high, you should exercise caution when using this parameter on a networked PC running NetWare.

  QEMM's DOS-Up feature normally loads your command processor into upper memory by adding LOADHI.COM to your SHELL statement in the CONFIG.SYS file. With the LOADHI.COM syntax added, your SHELL statement would look something like this (the entire command would appear on a single line and the n in R:n would be replaced by a High RAM region number):

  SHELL=C:\QEMM\LOADHI.COM /R:n
  C:\DOS\COMMAND.COM C:DOS\ /E:2048 /P

  DOS's default environment size is 160 bytes. If you need more conventional memory (and, especially if you are using the /E: switch on your SHELL line to create a larger environment, as in the above example) you may want to add the /ENVHI parameter to have QEMM load the DOS environment into upper memory as follows:

  SHELL=C:\QEMM\LOADHI.COM /ENVHI /R:n
  C:\DOS\COMMAND.COM C:\DOS\ /P /E:2048

  In the above example, the /ENVHI parameter would gain you an extra 2K of conventional memory by loading the environment into upper memory. (In the example above, the entire command would be typed on a single line.)

- **/EXCLUDELARGEST[:n] (/XL[:n])**
  tells LOADHI not to use the largest block (or the **n**th largest block) to load the target program.

- **/EXCLUDEREGION:n (/XR[:n])**
  tells LOADHI not to use region number **n** to load the target program.

- **/EXCLUDESMALLEST[:n] (/XS[:n])**
  tells LOADHI not to use the smallest block (or the nth smallest block) to load the target program.

- **/GETSIZE[:file] (/GS[:file])**

  tells LOADHI to report the amount of memory a program requires. With this option, LOADHI loads the program you specify and reports the following: 1) how many bytes the program took to initialize; 2) how many bytes the program occupies when resident; 3) whether the program can use LOADHI's Squeeze feature (see the SQUEEZEF and SQUEEZET parameters below); 4) the amount of upper memory the program used to initialize; and 5) the amount of upper memory the program uses once resident. You use the GETSIZE option to obtain information to help you custom fit your device drivers and TSRs into High RAM.

  LOADHI /GETSIZE actually loads the program to obtain the information. There are some programs that behave differently if another copy of the program is already loaded; so, if you want statistics on loading one copy of the program, be sure the program is not already loaded at the time you type this command. Also, the LOADHI /GETSIZE command does not automatically load the program into upper memory; the program will only load into upper memory if it has its own parameters that tell it to load there.

  The optional variable, **file**, specifies a filename. When a filename is present, LOADHI writes the program name, the size values and other information into this file so that you can examine it later. The information is written to the file as follows:

/I=	initialization size in conventional memory
/R=	resident size in conventional memory
/HI=	initialization size in High RAM
/HR=	resident size in High RAM
/M	program is a mouse driver
/SH	identifies the program as a shell
/UT=	a TSR's Squeeze information readable by Optimize
/UD=	a device driver's Squeeze information readable by Optimize
/F	OK to SQUEEZEF the program
/NSQ	not OK to Squeeze this program
/TSR=	size that LOADHI.COM /TSR will leave resident
/L=	unique identifying label

If the filename you specify already exists, LOADHI appends the information to this file. By systematically using the GETSIZE option with a filename you can compile the memory requirements of all the programs you wish to relocate to High RAM. This is the procedure that the Optimize program uses.

- **/HAPPIEST (/H)**

  tells LOADHI to use the smallest block of memory in which the program will fit, provided that it does not terminate with an error. Both device drivers and TSRs return after initializing, and the LOADHI programs can determine if the load was successful. If it was not, then LOADHI will try again with a larger area until the program exits successfully. If necessary the program will be loaded in conventional memory.

- **/HELP**

  displays a list of the LOADHI parameters and a brief description of each one.

- **LABEL:nn (/LB:nn)**

  is an internally used parameter.

- **/LARGEST[:n] (/L[:n])**

  tells LOADHI to load the program into the largest available block or a particular block out of several large blocks available. The number indicated by **n** indicates which block to use. For instance, the option /L:2 specifies the second largest available block.

- **/LINKTOP (/LINK)**

  connects free memory in High RAM to the DOS memory chain. There are a very few programs (e.g., some versions of FoxPro) that can take advantage of upper memory linked to the DOS memory chain (because that memory is non-contiguous with conventional memory). Run LOADHI /LINK before loading an application which can use this extra memory. When using LINK, it must be the only parameter on the LOADHI command line—you cannot also specify a program to be loaded.

- **/LO**

  tells LOADHI to unconditionally use conventional memory instead of High RAM. You can use this parameter to temporarily change

LOADHI statements in CONFIG.SYS or AUTOEXEC.BAT files without removing them completely. See the **REGION** parameter.

- **/NOLO (/NL)**

  tells LOADHI to prevent the program from being loaded in conventional memory if it does not fit in a High RAM region. This allows you to specify device drivers or TSRs that you would like to use, but only if they will fit in High RAM.

- **/QUIET (/Q)**

  suppresses the display of non-fatal error messages when LOADHI loads a program.

- **/REGION:n (/R:n)**

  tells LOADHI to load the program into region number **n**. If you want to specify a region for a program that is capable of loading itself high (e.g., Hyperdisk, Super PC-Kwik, certain Norton Utilities programs) use LOADHI /LO /R:n.

- **/RESIDENTSIZE=nnnn[K] (/RES=nnnn[K])**

  is a parameter used internally by the Optimize program. It tells LOADHI the resident size of the program being loaded into High RAM. The size nnnn is expressed in bytes (e.g., 4096) or kilobytes (e.g., 4K). /RESIDENTSIZE is used in conjunction with the /SQF or /SQT parameter; it allows LOADHI to verify that there is enough memory left in the chosen region to safely load the program.

- **/RESPONSEFILE[:file] (/RF[:file])**

  tells LOADHI to look into a file to find the parameters for the program being relocated. /RESPONSEFILE may be followed by a colon and a filename (or pathname, if necessary). If you do not specify a file, LOADHI.COM will look for the environment variable LOADHIDATA to determine the name and location of the file; the default is \QEMM\LOADHI.RF. The LOADHIDATA environment variable is created when you run Optimize on a system that has an MS-DOS 6 multiple configuration or when you run OPTIMIZE with the RESPONSEFILE parameter.

  If you are using MS-DOS 6 multiple configurations, Optimize will place the /RF parameter instead of the /REGION (/R:n) parameter on LOADHI statements. The response file itself will contain the High

RAM region information (i.e., the /R:n parameter) for each driver being loaded in each configuration path.

- **/SHELL**

  tells LOADHI to load a command processor into upper memory. Normally, QEMM's DOS-Up feature loads the command processor into upper memory without any special steps required. The /SHELL parameter is designed for two special cases: 1) To load an additional copy of a command processor into upper memory, or 2) To load a command processor high if something has occurred to prevent LOADHI from recognizing that it is loading a command processor (the symptom of this would be LOADHI consuming all of conventional memory, so you would not be able to run applications). The /SHELL parameter only works with LOADHI.COM.

- **/SIZE:nnnn[K]**

  tells LOADHI to allocate from a block that will best fit the size **nnnn**, where **nnnn** is a number expressed in bytes (e.g., 4096) or in kilobytes (e.g., 4K). The number you supply may come from the report issued by the GETSIZE option, or a number that you have determined by other means. In either case this number should represent the amount of memory the program needs to successfully initialize.

- **/SMALLEST[:n] (/S[:n])**

  tells LOADHI to load the program into the smallest available block or one particular block out of several small blocks. The number indicated by **n** indicates which of these to use. For instance, the option /S:2 specifies the second smallest block.

- **/SQUEEZEF (/SQF)**

  is an internally used parameter generated by the Optimize program. It instructs LOADHI to make temporary use of the page frame to give a device or TSR program sufficient room to initialize. The target program must not require the presence of this additional memory after it becomes resident. See also the /SQT parameter below. QEMM's Stealth ROM feature uses the page frame, so if Stealth ROM is enabled, LOADHI's SQUEEZEF parameter will be ignored.

- **/SQUEEZET (/SQT)**

  is an internally used parameter generated by the Optimize program. It instructs LOADHI to make temporary use of an area of upper memory for the purpose of giving a device or TSR program sufficient

room to initialize. The target program must not require the presence of this additional memory after it becomes resident. See also the /SQF parameter above.

- **/STUB**

  is a troubleshooting parameter you can use if you experience problems loading a device driver with LOADHI.SYS, or using the driver once it is loaded. If you are experiencing these kinds of problems with a particular device driver, especially if you had no such problems with a version of QEMM earlier than 7, try adding /STUB to the appropriate LOADHI.SYS line in CONFIG.SYS (e.g., DEVICE=C:\QEMM\LOADHI.SYS /STUB /R:n device_driver_path). /STUB causes a "placeholder" to be left in conventional memory when the device is loaded into upper memory. The placeholder contains the actual address of the device driver and some other information.

- **/TERMINATERESIDENT (/TSR)**

  tells LOADHI to terminate as a TSR, leaving a small stub of code (about 100 bytes) resident. This option is only useful if you are using LOADHI in combination with DOS's INSTALL command in CONFIG.SYS. In this case, the effect of this option is simply to suppress an error message issued by DOS indicating a failure to load a TSR when in fact LOADHI successfully relocated the program.

- **/UNLINKTOP (/UNLINK)**

  disconnects free memory in High RAM from the DOS memory chain. When using UNLINK, it must be the only parameter on the LOADHI command line. See the /LINK parameter above.

- **/?**

  displays a list of the LOADHI parameters.

LOADHI Notes for MS-DOS 5 and 6

Optimize will replace DOS DEVICEHIGH and LOADHIGH statements with the equivalent LOADHI commands in your system configuration files. However, if you have not run Optimize, and you have programs loaded in High RAM using the DEVICEHIGH or LOADHIGH commands, these programs will be included in the reports produced by LOADHI.COM and Manifest just as if they had been placed there by either of the LOADHI programs.

NOTES:

9 QEMM.COM:
Reports, Analysis and Mode Switching

This chapter describes QEMM.COM, a program that reports on how QEMM is managing your PC's memory. You can also use QEMM.COM to change QEMM's mode (e.g., ON or OFF), to help you diagnose and correct memory conflicts, and to display your QEMM registration information. Read this chapter if you are interested in finding out about QEMM's memory management on your PC, or if you want to diagnose a possible memory conflict or change QEMM's state.

You can use the QEMM.COM to:

- Change QEMM's mode (state) to ON, OFF or AUTO.
- Report status information about QEMM and information about the first megabyte of memory. Some of the information displayed by QEMM.COM is the same as that displayed in Quarterdeck Manifest's QEMM category.
- Report on what memory your programs have accessed and recommend additional upper memory addresses that you can include for use as High RAM. You can also use QEMM.COM to help resolve memory conflicts that may occur when a program or hardware device needs access to a specific upper memory area that is being used as High RAM. In this case, QEMM.COM will recommend what addresses to exclude from being used as High RAM.
- Display your QEMM registration information.

QEMM.COM commands consist of **QEMM** followed by a parameter. Some parameters have an abbreviation you can use instead of the name. The abbreviation is shown in parentheses following the parameter name.

To get an on-screen list of QEMM.COM's parameters:

- **Type QEMM ? and press Enter↵ to see a list of the names and abbreviations of all QEMM.COM parameters.**

or **Type QEMM HELP and press Enter ↵ to see a list of the parameters with a short description of what they do.**

Changing QEMM's Mode

There may be times when you want to change QEMM's current mode. QEMM has the following modes:

- **AUTO (AU)** - QEMM will turn ON when a program requests expanded memory or any service that puts QEMM into virtual-8086 mode.
- **ON** - Expanded memory is available and the processor is in virtual-8086 mode. The mode is forced ON if any of the following conditions are met: High RAM is created, ROMs are mapped into RAM, conventional or video memory is filled, conventional memory is sorted, expanded memory is in use or Stealth ROM is enabled. During QEMM's installation, the RAM parameter is added to the QEMM386.SYS line of your CONFIG.SYS file; that parameter forces the mode to ON.
- **OFF** - Expanded memory is unavailable and the processor is in real mode.

To change the mode, type QEMM followed by the mode. For example, to change the mode to ON:

- **Type QEMM ON and press Enter ↵.**
- **You can similarly change the mode to OFF or AUTO, provided there are no conditions that have forced the mode ON.**

QEMM.COM Reports

QEMM.COM has five reports which give you information you can use to optimize your memory configuration. To see a report, you type QEMM followed by the report name. These five reports are:

- **Summary (SUM)** - This report tells you QEMM's current mode, the amount of expanded memory currently available, the page frame address and which Stealth ROM method (if any) is enabled.
- **Type (T)** - This report tells you how QEMM is managing the first megabyte of memory.

- **Accessed (ACC) -** This report indicates the areas of memory that have and have not been accessed from the time QEMM started running (or you did a QEMM RESET) until the time of the display.

- **Analysis (AN) -** This report cross-references the information in the Accessed and Type reports. Based upon what memory has been specified and how QEMM is configured, Analysis suggests command line parameters you can add to the QEMM386.SYS device driver line in CONFIG.SYS to use upper memory more efficiently. You should use these suggestions only after following the Analysis procedure explained later in this chapter.

- **Memory (MEM)** - This report gives you an accounting of the memory in your PC (conventional, High RAM, extended, and expanded) both before and after QEMM386.SYS has configured this memory.

The Type, Analysis, and Accessed reports have an optional MAP format which gives you a block diagram of memory. To use the MAP format just include the MAP parameter on the command line. For example, to display the Type report as a map, you would type **QEMM TYPE MAP**.

If you type QEMM without any parameters, you will get the Summary report followed by the Type report.

The Summary Report

The Summary report lists basic status information about QEMM. To see a Summary report:

- **Type QEMM SUMMARY and press Enter ↵.**

The Summary report displays.

- **Current Mode** is QEMM's mode: ON, OFF, or AUTO.

- **Expanded Memory Available** is how much EMS memory is available.

- **Page Frame Address** gives the address of the page frame. This entry indicates "none" if there is no page frame.

- **Stealth Type** indicates the Stealth ROM method in use, if any. M is the mapping method and F is the frame method.

```
C:\>qemm summary

Current Mode                = ON
Expanded Memory Available   = 4640K
Page Frame Address          = C000H
Stealth Type                = M
Stealth ROMs                = F000: 64K
                              C000: 32K

Expanded memory is being used.
```

The Summary report

- **Stealth ROMs** indicates the starting address and size of any Stealthed ROMs.

The Type Report

The Type report offers two views of the first megabyte of memory as managed by QEMM. The default view is a list—showing memory areas, their size and use. The second view is a map, showing a block diagram of the first megabyte. To see the Type report:

- Type **QEMM TYPE** or **QEMM TYPE MAP** and press **Enter** ↵.

```
C:\>qemm type

    Area        Size      Status
0000 - 0FFF      64K    Excluded
1000 - 9FFF     576K    Mappable
A000 - AFFF      64K    Video
B000 - B7FF      32K    Excluded
B800 - BFFF      32K    Video
C000 - CFFF      64K    Page Frame
D000 - FFFF     192K    High RAM
```

The Type report

Key terms used in the Type reports are:

- **Mappable (+):** Memory addresses that can be filled with memory using EMS 4.0 and EEMS function calls. Mappable areas must be 16K and aligned on 16K boundaries. QEMM386.SYS's RAM

parameter automatically converts mappable upper memory areas to High RAM.

- **Rammable** (*): Memory areas that QEMM can fill with High RAM but are too small to be accessed by EMS function calls (i.e., less than 16K in size). QEMM can convert rammable areas to High RAM.

- **Page Frame (F)**: A 64K mappable area that EMS programs use for expanded memory access.

- **High RAM (H)**: Areas between 640K and 1024K that QEMM has filled with RAM. QEMM will convert Mappable or Rammable areas to High RAM if the RAM parameter is present on the QEMM386.SYS line in CONFIG.SYS.

- **Mapped ROM (M)**: ROM that has been mapped to RAM by QEMM386.SYS's ROM parameter. Mapping ROMs lets ROM code run in faster RAM. When Stealth ROM is not enabled, QEMM automatically maps a 4K piece of system ROM somewhere between F000 and FFFF to detect warm reboots.

- **Excluded (X):** Memory addresses explicitly made unavailable for High RAM, expanded memory mapping, or ROM mapping by QEMM386.SYS's EXCLUDE parameter. If QEMM does not automatically detect an area specifically needed by an adapter or a program, you must explicitly exclude the area. On 386-based PCs only, QEMM automatically excludes the addresses from 0000 to 0FFF to prevent problems that could arise due to a bug in some 386 processors.

- **Video (V):** Addresses reserved for video display memory.

- **Adapter RAM (A):** Addresses that have RAM placed into them by adapter cards (e.g., a network adapter).

- **ROM (R):** ROM areas not remapped by QEMM386.SYS's ROM parameter.

- **Split ROM (/)**: Addresses that have ROM which occupies only a part of the 4K area. These areas cannot be Stealthed or remapped by QEMM386.SYS's ROM parameter.

In the map, each 16-character row represents 64K of memory. Each character represents 4K of memory, and has a special meaning

(described above). The left column shows addresses (0n00 through Fn00). To find a particular 4K area, replace the letter *n* with the hex number at the top of the table.

```
C:\>qemm type map

    n=0123 4567 89AB CDEF
0n00 XXXX XXXX XXXX XXXX
1n00 ++++ ++++ ++++ ++++
2n00 ++++ ++++ ++++ ++++
3n00 ++++ ++++ ++++ ++++
4n00 ++++ ++++ ++++ ++++
5n00 ++++ ++++ ++++ ++++
6n00 ++++ ++++ ++++ ++++
7n00 ++++ ++++ ++++ ++++
8n00 ++++ ++++ ++++ ++++
9n00 ++++ ++++ ++++ ++++
An00 VVVV VVVV VVVV VVVV
Bn00 XXXX XXXX VVVV VVVV
Cn00 FFFF FFFF FFFF FFFF
Dn00 HHHH HHHH HHHH HHHH
En00 HHHH HHHH HHHH HHHH
Fn00 HHHH HHHH HHHH HHHH

+ = Mappable
* = Rammable
F = Page Frame
H = High RAM
M = Mapped ROM
X = Excluded
V = Video
A = Adapter RAM
R = ROM
/ = Split ROM
```

The Type report (map)

The Accessed Report

The Accessed report offers two views (list and map) of the first megabyte of memory, indicating the 4K regions of memory that have been accessed since QEMM started or since you typed QEMM RESET (see below). You can use this report to see how much memory a program uses and whether it accesses upper memory. This report is valid only if QEMM's mode is ON. To see the Accessed report:

- **Type QEMM ACCESSED or QEMM ACCESSED MAP and press Enter ↵.**

```
C:\>qemm accessed

    Area        Size      Status
0000 - 28FF     164K   Accessed
2900 - 2CFF      16K   Unaccessed
2D00 - 2DFF       4K   Accessed
2E00 - 95FF     416K   Unaccessed
9600 - B7FF     136K   Accessed
B800 - BBFF      16K   Unaccessed
BC00 - BFFF      16K   Accessed
C000 - CFFF      64K   Unaccessed
D000 - FFFF     192K   Accessed
```

The Accessed report

- **Unaccessed** means an area has not been read or written by a program. Unaccessed areas above 640K can potentially be filled with High RAM.

- **Accessed** areas have been accessed by a program. In the Accessed report's map format, **Accessed** means the area has been read by a program and **Written** means a program has written to the area.

```
C:\>qemm accessed map

    n=0123 4567 89AB CDEF
0n00 WWWA AAAA AAAW WWWW
1n00 WWWW WWWW WWWW WWWW
2n00 WWWW WWWW WUUU UWUU
3n00 UUUU UUUU UUUU UUUU
4n00 UUUU UUUU UUUU UUUU
5n00 UUUU UUUU UUUU UUUU
6n00 UUUU UUUU UUUU UUUU      U = Unaccessed
7n00 UUUU UUUU UUUU UUUU      A = Accessed
8n00 UUUU UUUU UUUU UUUU      W = Written
9n00 UUUU UUWW WWWW WWWW
An00 WWWW WWWW WWWW WWWW
Bn00 AAAA AAAA UUUU WWWW
Cn00 UUUU UUUU UUUU UUUU
Dn00 WAAW WWWW WWWW WWWW
En00 WWWW WWWW WWWW WWWW
Fn00 WWWW WWWW WWWW WWWW
```

The Accessed report (map)

The Accessed report may be inaccurate for mappable areas. When QEMM maps EMS memory into a mappable area, it clears the area's state, making it appear to be unaccessed. Active use of the page frame for mapping will also clear the "accessed indicators."

Resetting Memory

You can reset the state of memory to Unaccessed by typing **QEMM RESET**. You may want to reset memory before running a program to help determine which memory areas the program accesses.

The Analysis Report

As you run programs, QEMM keeps a record of the memory they access. The Analysis report evaluates which memory is available for High RAM or expanded memory mapping and tells you if there are particular areas you should exclude from QEMM's management (e.g., adapter RAM or ROM that was not detected by Optimize). The Analysis report will also tell you if there are additional areas you can use as High RAM.

The Analysis report is a useful troubleshooting device if you have a particular program that does not run or display properly after you install QEMM. The Analysis report can tell you if the program in question is attempting to access an upper memory area that QEMM can fill with High RAM. If this turns out to be the case, you can exclude that area from QEMM's management. You must follow the procedure below for the Analysis report to be accurate.

1. The first step is to determine if QEMM's Stealth ROM feature is enabled and, if so, which Stealth ROM method, F or M is in use. At the DOS prompt, type **QEMM** and press **Enter** ↵.

   A report summarizing QEMM's status will display. If you see the line Stealth Type = M or Stealth Type = F, Stealth ROM is enabled; remember which letter is listed, M or F.

2. Use a text editor to edit your CONFIG.SYS file and type REM followed by a space at the beginning of the line that starts as follows:

   DEVICE=C:\QEMM\QEMM386.SYS (REM causes the line to be ignored).

   The line should look something like this:

   REM DEVICE=C:\QEMM\QEMM386.SYS (parameters not shown)

3. Add a new line directly below the line you just edited. What the line says depends on whether Stealth ROM is enabled on your system.

   If you are using Stealth ROM: Add the following line, substituting the appropriate Stealth ROM letter, M or F for the x in ST:x.

   DEVICE=C:\QEMM\QEMM386.SYS ON MAPS=0 ST:x

   If you are not using Stealth ROM: Add the following line:

   DEVICE=C:\QEMM\QEMM386.SYS ON MAPS=0

4. Save your CONFIG.SYS file and reboot your PC.

5. What you do next depends on what kind of analysis you want to do.

   Partial Analysis: If you are troubleshooting a program that does not display or run properly, or a hardware operation that does not work properly (e.g., a floppy drive), you will want to do a partial analysis, which simply entails using the program or device that has been giving you trouble since you installed QEMM.

   To do a partial analysis, just run the programs that have not been working and access their basic features. Or, if you are having trouble with a particular hardware operation, perform that operation. When you are done, skip to step 6.

   Comprehensive Analysis: If you want to fully analyze your system and programs to see if you can gain additional High RAM, you will want to do a *comprehensive analysis*. This kind of analysis is somewhat time-consuming—you will need to run all your programs and access all hardware devices.

   To do a comprehensive analysis, run your programs and use their basic functions (be sure to access any graphics-based programs). If

you use DESQview, run its programs and features, then quit DESQview. Do not use any hardware or memory analysis utilities other than QEMM. Access all your PC's hardware and peripherals. If you have two monitors, display information on both of them. Access all disk drives. Format a diskette. Print a document. If you have a network adapter card or terminal emulation adapter, access the network or terminal emulator. When you are done, continue with step 6 below.

Be aware that the new QEMM line you added for the Analysis procedure did not include the RAM parameter. That means you no longer have High RAM, so device drivers and TSRs will load low and you will not have as much conventional memory to run programs. We will add the RAM parameter back later.

Special note to DESQview and DESQview/X users: DESQview and DESQview/X use available portions of upper memory for themselves. If you run an Analysis procedure to find out what upper memory areas a program accesses when running under DESQview or DESQview/X, the results of the Analysis procedure will be misleading—DESQview's use of upper memory will prevent you from seeing accesses by the program in question. You can get around this by running DESQview or DESQview/X with a parameter that prevents DESQview itself from using upper memory. To do that, start DESQview by typing DV /X:A000-FFFF (or DVX /X:A000-FFFF for DESQview/X). Be aware that DESQview will be using more conventional memory since you are preventing it from using upper memory. That means there will be less memory for each window. If your program requires too much memory to fit in the smaller window, you cannot run the program under DESQview, so the Analysis will not help. If this is the case and you are troubleshooting a program, try changing the program's protection level in the program's DVP. See your DESQview or DESQview/X manual for information on changing a DVP.

6. Display the Analysis report by typing **QEMM ANALYSIS** or **QEMM ANALYSIS MAP** and pressing **Enter** ↵. If you are unfamiliar with hexadecimal addressing, we suggest using **QEMM ANALYSIS** without MAP—the report will be easier to interpret.

   The default view is in list format which explicitly lists addresses you may instruct QEMM386.SYS to include or exclude.

   This report uses three terms:

   OK (O): Memory areas QEMM has properly identified.

```
C:\>qemm analysis

    Area        Size      Status
0000 - AFFF     704K    OK
B000 - B7FF      32K    Exclude
B800 - C7FF      64K    OK
C800 - F7FF     192K    Exclude
F800 - FFFF      32K    OK

C:\>
```

The Analysis report (list)

Exclude (X): Areas of memory that have been accessed, but that QEMM expected would remain unaccessed. You should use the QEMM386.SYS's EXCLUDE parameter to prevent QEMM from using these areas.

Include (I): Upper memory areas that are reserved by QEMM but have not been used by a program. You may use QEMM386.SYS's INCLUDE parameter to allow QEMM to use these areas.

```
C:\>qemm analysis

   n=0123 4567 89AB CDEF
0n00 0000 0000 0000 0000
1n00 0000 0000 0000 0000
2n00 0000 0000 0000 0000
3n00 0000 0000 0000 0000
4n00 0000 0000 0000 0000
5n00 0000 0000 0000 0000
6n00 0000 0000 0000 0000        0 = OK
7n00 0000 0000 0000 0000        X = Exclude
8n00 0000 0000 0000 0000        I = Include
9n00 0000 0000 0000 0000
An00 0000 0000 0000 0000
Bn00 XXXX XXXX 0000 0000
Cn00 0000 0000 XXXX XXXX
Dn00 XXXX XXXX XXXX XXXX
En00 XXXX XXXX XXXX XXXX
Fn00 XXXX XXXX 0000 0000
```

The Analysis report (map)

7. Examine the Analysis report and jot down the suggestions. Edit CONFIG.SYS and delete the line you added earlier. It looks like one of the following:

   DEVICE=C:\QEMM\QEMM386.SYS ON MAPS=0 ST:x

   or

   DEVICE=C:\QEMM\QEMM386.SYS ON MAPS=0

8. Delete the word REM you added at the beginning of the original QEMM386.SYS line. Then, use the EXCLUDE or INCLUDE parameters to exclude or include any memory addresses specified in

the Analysis report. You can abbreviate EXCLUDE as X, and INCLUDE as I. For example, if the Analysis report recommended excluding B000-B7FF, you would add EXCLUDE=B000-B7FF to the end of the QEMM386.SYS line. For information on the INCLUDE and EXCLUDE parameters, see **Chapter 7**.

9. Save the CONFIG.SYS file, reboot your computer and rerun Optimize (see **Chapter 3** for information on Optimize).

The Memory Report

You can use the Memory report to see how QEMM has configured your PC's memory. The Memory report gives information on conventional memory, High RAM, extended and expanded memory, before and after QEMM has configured memory. For some PCs there will be another category: top memory (on Compaq PCs and Compaq compatibles) or Shadow RAM (on PCs using the Chips & Technologies chip set or other shadow memory supported by QEMM). To see a Memory report:

- Type **QEMM MEMORY** and press **Enter** ↵.

```
C:\>qemm memory

                          Unavailable  Converted
              Initial     to QEMM      by QEMM     Leaving
Conventional:   640K   -     0K    -     0K    =  640K
Extended:     16640K   -     0K    -16640K    =    0K
Expanded:         0K   -     0K    +16288K    =16288K
High RAM:         0K   -     0K    +  192K    =  192K
              ------     ------     ------     ------
   TOTAL:     17280K   -     0K    -  160K    =17120K

                    160K QEMM Overhead
     Code & Data:    119K     Maps:              0K
     Tasks:           18K     Mapped ROM:        0K
     DMA Buffer:      12K     Unassigned:       11K
          4.1K Conventional Memory Overhead

C:\>
```

The Memory report

- The **Initial** column summarizes your PC's memory before QEMM loads. The **Expanded** and **High RAM** categories always start at 0K. The column's total should equal the amount of your PC's memory.

- The **Unavailable to QEMM** column shows any memory QEMM cannot manage. Device drivers that use extended memory and load before QEMM386.SYS make that memory unavailable to QEMM. Memory excluded from QEMM's management by

QEMM386.SYS's EXTMEM or MEMORY parameter will also be listed as unavailable. PCs with shadow memory make some of that memory unavailable to QEMM.

- The **Converted by QEMM** column shows QEMM converting extended memory (and shadow memory or Top memory, if any) into expanded memory and High RAM. QEMM can fill in missing conventional memory, so you may find it creates more conventional memory. This column's total shows how much memory QEMM keeps for its own use and for mapping ROMs.

- The **Leaving** column shows how much memory is left once QEMM initializes. **Conventional** shows the available contiguous memory for DOS programs. **Extended** (usually 0K) shows how much extended memory is left for programs that access extended memory through the obsolete BIOS interface (e.g., VDISK). Programs that can access memory through EMS, XMS, VCPI or DPMI do not use this interface. **Expanded** shows how much memory is available to programs which support EMS, XMS, VCPI, or DPMI. **High RAM** is the amount of upper memory available for loading TSRs and device drivers.

- The lower part of the Memory report details how QEMM itself uses memory. At the bottom of the report is a number showing how little conventional memory QEMM uses.

- The **Unassigned** category of this report represents memory that QEMM controls but which has not been given any function. After QEMM has set up its data, High RAM and other resources, leftover memory is converted to expanded memory. Since expanded memory is allocated in 16K units, any memory smaller than 16K, or fragmented memory, is left unassigned. You can alter some of the QEMM386.SYS parameters (e.g., RAM, ROM) to increase or decrease the amount of unassigned memory, either to use it for these new purposes, or to increase it until it becomes more expanded memory.

If your PC has more memory than the report shows: Be sure your PC's CMOS is set up properly, and that there are no defective or loose memory chips. Also see the USERAM parameter in **Chapter 7**.

Why QEMM may not be able to use all your shadow memory: QEMM cannot use shadow memory which overlaps ROM or adapter or video RAM.

Let QEMM allocate extended memory: We suggest you load device drivers that use extended memory after QEMM386.SYS. Most extended memory programs can access memory through QEMM's XMS, VCPI or DPMI support. If your extended memory program does not use XMS, VCPI or DPMI (e.g., VDISK), use QEMM386.SYS's EXTMEM parameter (described in **Chapter 7**) and load the program after QEMM386.SYS.

Displaying Your Registration Information

To display your serial number and other QEMM registration information:

- Type **QEMM REG** and press **Enter** ↵.

NOTES:

10 QDPMI: QEMM's DPMI Host

This chapter is a technical reference for Quarterdeck's DPMI (DOS Protected Mode Interface). Most users will not need to know anything about QDPMI. This chapter is provided for advanced users who would like technical information about QDPMI.

The Quarterdeck DPMI Host (QDPMI) is a full implementation of Version 0.9 of the DOS Protected Mode Interface (DPMI) specification, including MS-DOS extensions and virtual memory. DPMI is an emerging industry standard for allowing DOS applications to access the protected mode features of 286 and higher processors. The DPMI specification was developed by a consortium of computer hardware and software companies, including Quarterdeck, Borland, Ergo, IBM, Intel, Intelligent Graphics, Locus, Lotus Development, Microsoft, Phar Lap, Phoenix Technologies and Rational Systems.

QDPMI is automatically installed on your system when you install QEMM.

Protected mode multitasking environments, memory managers, or operating systems that implement DPMI functions are called DPMI hosts. Protected mode applications that request DPMI functions (directly or indirectly) are called DPMI clients.

DOS applications using DPMI (DPMI clients) can run in a variety of operating environments, including DOS, DR DOS, Microsoft Windows, OS/2, 386-based UNIX, DESQview 386 and DESQview/X, provided a DPMI host is present.

The Quarterdeck DPMI Host (QDPMI) provides mode-switching services and extended memory management to DPMI clients. It runs at a more privileged level than its clients and uses the hardware to enforce a supervisor/user protection model. This allows QDPMI to support virtualization of memory, maintaining full control over client programs' address spaces.

Quarterdeck's DESQview 386 and DESQview/X can run multiple QDPMI clients simultaneously, each optionally having its own virtual memory.

Software Compatibility

The major DOS applications that require DPMI support are programmer's tools, including Microsoft's C/C++ Development System for Windows (version 7), Borland's C/C++ (version 3) and Intel's Code Builder Kit (version 1.1). QDPMI is compatible with all of these programs.

Microsoft Windows version 3.1 includes its own DPMI host. When QDPMI detects the startup of Microsoft Windows 386 enhanced mode, it allows Windows to provide the DPMI services for programs running in its environment.

QDPMI is automatically installed when you install QEMM. QEMM must be present for QDPMI's services to be made available to DPMI clients.

By default, QEMM's installation sets up a virtual memory swapfile on your hard disk for times when there is not enough physical memory available for DPMI clients. The swapfile starts out as zero-length and grows as needed up to the default maximum size of one megabyte. You can change the maximum swapfile size or specify that there should be no swapfile. To modify the swapfile option:

- **At the DOS prompt, type QSETUP.**
- **Press Enter ↵ at the Setup program's opening screen to go to the Setup menu.**
- **At the Setup menu, select Enable or disable DPMI Host.**
- **Follow the on-screen instructions.**

QEMM's Setup makes the necessary changes to your CONFIG.SYS file.

If you change your configuration, we suggest you run QEMM's Optimize program to ensure the best use of your system's memory (for information on Optimize, see **Chapter 3**).

To see what DPMI services are enabled:

- **At the DOS prompt, type QDPMI.**

QDPMI Client Initialization with Virtual Memory

When you start an application that requires DPMI services, QDPMI will attempt to create the virtual memory swapfile whose name and maximum size are specified as parameters to the QDPMI device driver line in CONFIG.SYS (see *QDPMI Parameters* later in this chapter), or optionally in the QDPMI environment variable. If the

swapfile name is not specified in either of these locations, a swapfile with the base name QDPMI.SWP is opened or created in the directory from which QDPMI.SYS was loaded (normally, \QEMM).

If the swapfile already exists, it will be truncated to zero length, and increased as needed by QDPMI.

If you specify the KILLSWAP option (see QDPMI Parameters," later in this chapter), QDPMI will delete the swapfile when the DPMI client terminates.

In DESQview or DESQview/X, each task or DESQview window can have its own QDPMI options in force. For information, see "Using QDPMI with DESQview or DESQview/X" later in this chapter.

Each simultaneously executing DPMI client utilizing virtual memory has its own virtual memory swapfile. The name of the swapfile provided by the QDPMI device driver's SWAPFILE parameter in CONFIG.SYS (or by the QDPMI environment variable) is therefore modified by QDPMI before it is opened or created, by appending a digit to the end of its base name. For example: QDPMI.SWP would become QDPMI0.SWP, and QDPMISWP.SWP would become QDPMISW0.SWP. The actual digit will vary depending on whether DESQview is active or not, and how many other DPMI clients are active in other DOS partitions.

If QDPMI cannot find or create its swapfile, it will issue a message and disable the virtual memory feature for the current client's invocation. Causes for this may be:

- Insufficient disk space.
- Improperly specified swapfile name (SWAPFILE parameter to QDPMI.SYS or environment variable).
- Non-existent directory specified.

System Interrupt Stack Configuration

For some applications, it is important that a line exist in CONFIG.SYS increasing the number of DOS hardware interrupt stacks. The suggested values are:

stacks = 9,256

This means that DOS should allocate nine system stacks, each 256 bytes in size. The default for this setting is stacks = 9,128 which may not be sufficient for some programs.

Novell LAN WorkPlace for DOS

QDPMI supports 16- and 32-bit DPMI clients. When a 32-bit client is executing, it expects all 32 bits of the processor's 32- bit registers to be preserved across interrupts. Some device drivers and TSRs utilize the processor's 32-bit registers, but do not preserve the high-order 16 bits, causing 32-bit DPMI clients to behave erratically. A notable example is Novell's LAN WorkPlace for DOS versions 4.0 and 4.01 (TCPIP.EXE). You can prevent the erratic behavior by running the supplied TSR LWPFIX.COM (in the QEMM directory) after loading the LAN WorkPlace TCPIP driver. Novell has fixed this problem in versions 4.10 and later.

Borland C/C++ 3.1

Borland's compiler can allocate memory through both EMS and DPMI APIs. QDPMI cannot limit EMS allocation by its clients, so it is possible for QDPMI's memory resources to be depleted without QDPMI's knowledge. To prevent this occurrence, use Borland's compile-time switch –QE=nnnn or –QE–. The former tells Borland's compiler not to exceed nnnn K of EMS memory for use as a swap buffer. The latter tells the compiler to use NO EMS for swapping purposes when compiling. The recommended setting is –QE–, as all necessary swapping will be performed by QDPMI.

QDPMI Parameters

This section is the reference guide for optional parameters used with both QDPMI.SYS and QDPMI.COM. This chapter also explains the way to use the optional QDPMI environment variable to control QDMPI's behavior.

Both QDPMI.SYS and QDPMI.COM respond to the same set of command line parameters. Command line parameters may be used on the QDPMI device driver line in CONFIG.SYS to initially set QDPMI's options. QDPMI.COM may also be used to modify any or all of QDPMI's options from the DOS command line at a later time.

To use a QDPMI.SYS command line parameter, you type the parameter name on the same line as DEVICE=C:\QEMM\QDPMI.SYS in your CONFIG.SYS file. To use a QDPMI.COM command line parameter you type QDPMI followed by the parameter name at the DOS prompt. You may use an abbreviation instead of the parameter name. The abbreviation for

each QDPMI command line parameter is shown below in parentheses after each parameter name.

- **EXTCHKOFF (NOXCHK)**
  causes the QDPMI host not to step out of the way for DOS extenders provided by Phar Lap, Rational Systems, and Ergo. These DOS extenders may not function properly if QDPMI is providing DPMI services.

- **EXTCHKON (XCHK)**
  causes the QDPMI host to step out of the way for DOS extenders provided by Phar Lap, Rational Systems, and Ergo; these DOS extenders prefer to manage DPMI in their own way. This is the default setting.

- **KEEPSWAP (KPSW)**
  prevents the QDPMI host from deleting its virtual memory swapfile upon termination of the DPMI client. This means that the same swapfile will be preserved on disk for future DPMI clients. This may take more disk space than KILLSWAP, but will save time.

- **KILLSWAP (KLSW)**
  causes the QDPMI host to delete its virtual memory swapfile upon the termination of the DPMI client. This is the default setting.

- **MAXLOW nnnnn (LOW nnnnn)**
  is the maximum number of kilobytes of conventional memory allowed by QDPMI for use as its memory resources. Setting it to any non-zero value will allow QDPMI to allocate conventional memory for use in its memory pool. QDPMI will use DOS conventional memory only if expanded and extended memory resources are insufficient to satisfy the MINMEM request. The default value is 0. Conventional memory is only used if MINMEM cannot be satisfied from other resources, and only if this value is not zero.

- **MAXMEM nnnnn**
  is the maximum number of kilobytes of physical memory that will be reserved by QDPMI. This can be any number up to the amount of physical memory available on your system, and will be allocated from VCPI and conventional memory resources, in that order. The memory used by QDPMI includes all memory for use by its code, data, and page tables, as well as memory requested for use by the

DPMI client application. If MAXMEM is greater than the available physical memory, MAXMEM will be adjusted downward to be within those limits. Similarly, if you are using DESQview or DESQview/X and MAXMEM is greater than the limits DESQview places on extended memory in a window, MAXMEM will be adjusted downward to be within those limits.

The default value is the total amount of physical memory installed in your system. If omitted or explicitly set to zero, MAXMEM is set to the largest value supported on your system. This will either be the size of all of your physical memory, or that which is available to DESQview's or DESQview/X's current partition, whichever is smaller.

- **MINMEM nnnnn**

  is the amount of physical memory in kilobytes that will be reserved by QDPMI immediately upon DPMI client initialization for use by the QDPMI kernel itself. QDPMI will initialize only if at least this amount of memory is available for its use.

  MINMEM's default value depends on the amount of physical memory installed in your computer, and whether you are using QDPMI's Virtual Memory features. Any attempt to set MINMEM to a value lower than QDPMI's calculated minimum will result in QDPMI ignoring the request, and using its own calculated MINMEM value.

  MAXMEM must be greater than or equal to MINMEM, regardless of whether it was user specified or internally calculated. If this is not the case, QDPMI will exit with a message indicating that insufficient memory is available for initialization.

  The difference between MAXMEM and MINMEM is the amount of memory that will be managed dynamically by QDPMI, and made available to other applications (including other DPMI clients) when it is free.

- **OFF**

  turns DPMI services off.

- **ON**

  turns DPMI 0.9 services on. This is a default setting.

- **OVLDIR=path**
  specifies where QDPMI will look for its QDPMIVM.OVL file during DPMI client initialization. This defaults to the same directory from which QDPMI.SYS was loaded at system initialization. Under some network implementations such as diskless workstations, this initial path may cease to exist after the system is initialized. Use this switch to tell QDPMI where to look for QDPMIVM.OVL at DPMI client initialization in such cases.

- **SWAPFILE name (SWAP name)**
  is the base name of the virtual memory swapfile that QDPMI will use.

- **SWAPSIZE nnnnn (SWSZ nnnnn)**
  is the virtual memory swapfile's size in kilobytes. The size of the virtual memory swapfile that you specify may be any value up to 64 megabytes, and defaults to 1MB.

- **VMOFF (NOVM)**
  turns QDPMI virtual memory off.

- **VMON (VM)**
  turns QDPMI virtual memory on. This is the default setting.

The QDPMI Environment Variable

You can optionally use an environment variable to determine several aspects of QDPMI's configuration. Its syntax is as follows:

SET QDPMI=[MAXLOW nnnnn] [MINMEM nnnnn]
[MAXMEM nnnnn] [SWAPFILE name] [SWAPSIZE nnnnn]

All of the options set in the environment variable are normally set by the parameters to the QDPMI device driver line (see above). If you choose to use the environment variable, its settings will override the corresponding settings in the QDPMI device driver line. You cannot use parameter abbreviations with the QDPMI environment variable.

Using QDPMI with DESQview or DESQview/X

If you are using DESQview or DESQview/X, you may want to have a different set of DPMI options in force for different windows. There are two ways to do this:

- Set a QDPMI environment variable for each window you want to affect, using the desired options (see the previous section). You can do this by typing the SET QDPMI= command at a window's DOS prompt or in a script.

- Run the QDPMI.COM program with the desired parameters in each window you want to affect. You can do this by typing the QDPMI command at a window's DOS prompt or in a script. The only parameters that are limited in scope to a particular window are ON, OFF, VMON, VMOFF, KILLSWAP, KEEPSWAP, EXTCHKON, and EXTCHKOFF. If you use any of the other QDPMI parameters, they will affect all windows. (For information on the parameters, see the section, "QDPMI Parameters," earlier in this chapter).

Virtual Memory and Total Memory Size

The sum of the value of MAXMEM and the maximum size of the virtual swapfile is the amount of virtual memory (in kilobytes) available to QDPMI client applications when the virtual memory option is enabled.

Error Messages

QDPMI may display error messages when trying to load itself or when trying to initialize a DPMI client. Error messages may also be generated by the QDPMI kernel or the DOS extender.

- **QDPMI: Client already active — Cannot modify parameters.**

  A DPMI client has spawned COMMAND.COM, and is still active in the current DOS partition. It is inappropriate to change QDPMI parameters while a QDPMI client is currently running. This condition can sometimes be caused by the abnormal termination of a DPMI client.

- **QDPMI: Already supported — Not reloaded.**

  A DPMI Host other than QDPMI has been detected. Quarterdeck's QDPMI is not loaded.

- **QDPMI: Version incompatibility error — Re-install or use same version.**

  This message only appears if the device driver QDPMI.SYS and the utility program QDPMI.COM are not the same version. This is an error in installation or updating. You must reinstall QDPMI.

- **QDPMI: Cannot locate QEMM386.SYS Memory Manager — Not loaded.**

  QDPMI Host will only function in conjunction with QEMM.

- **QDPMI: Error accessing XDI — Not loaded.**

  QDPMI has encountered an error initializing the DESQview XDI interface for external devices and cannot load.

- **QDPMI: Too Many DPMI Processes — Not loaded.**

  QDPMI can support up to fifteen concurrent processes under DESQview and DESQview/X. If a DPMI client is started in the 16th or higher window, the DPMI client will fail to initialize, and QDPMI.COM will display this message if run in this window.

- **QDPMI: Overlay File Path/Name is too long.**

  The QDPMIVM.OVL pathname is too long. Use a path less removed from the root.

- **QDPMI: Quarterdeck QDPMI Host is not resident.**

  QDPMI.COM issues this message if run without having previously loaded QDPMI.SYS in CONFIG.SYS.

QDPMI client initialization error messages are as follows:

- **QDPMI: Open Error.**

  QDPMI could not open its overlay file QDPMIVM.OVL. Be sure that it is either in the same directory as QDPMI.SYS, or in the directory specified by the OVLDIR parameter on the QDPMI command line.

- **QDPMI: Processor Error.**

  QDPMI must have an 80386 or higher processor in order to run.

- **QDPMI: Read Error.**

  QDPMI cannot read its overlay file QDPMIVM.OVL.

- **QDPMI: Seek Error.**

  QDPMI cannot seek in its overlay file QDPMIVM.OVL.

- **QDPMI: Close Error.**

  QDPMI cannot close its overlay file QDPMIVM.OVL.

- **QDPMI: Bad .OVL Error.**

  QDPMIVM.OVL file is corrupted.

- **QDPMI: Exec Error.**

  QDPMI could not transfer control to its overlay.

- **QDPMI: Init Error.**

  QDPMIVM.OVL could not complete its initialization.

- **QDPMI: Mode Error.**

  The QDPMI client is attempting to re-initialize the QDPMI Host in a mode (16/32 bit) other than the one in which it was originally initialized. This can occur if an attempt is made to start a DPMI client of another "bitness" than the currently running DPMI client.

- **QDPMI: Task Error.**

  The maximum of fifteen simultaneous DPMI clients are already running in other DESQview or DESQview/X windows.

- **QDPMI: Version Error.**

  There is a discrepancy between the QDPMI version of the device driver (QDPMI.SYS), and the kernel (QDPMIVM.OVL).

11 EMS Utility Programs

This chapter describes three advanced EMS utility programs that can give you more control over EMS memory. This chapter is provided for programmers and advanced users who want to control EMS memory allocation.

QEMM includes three advanced utility programs: EMS.COM and EMS.SYS, which give you behind-the-scenes access to QEMM's management of EMS memory; and EMS2EXT, which lets you allocate extended memory on the fly through the pre-XMS INT 15 interface. You may never need these programs, but they can prove useful in cases where you need more control of memory allocation than your programs give you.

The EMS Programs

The EMS.COM and EMS.SYS programs provide several informative and powerful functions to help you make the best use of your EMS memory in cases in which you have special or unusual requirements. Although anyone may benefit from seeing the EMS status report and the details of expanded memory allocation, other uses of EMS which will be described in these sections are for technically sophisticated users.

Most of the functions of EMS.SYS and EMS.COM involve the manipulation of expanded memory handles. An *EMS handle* is the information that the expanded memory manager uses to identify a block of memory that it allocates. A handle is represented by a number and may optionally have a name.

An expanded memory handle is the token of interaction between an EMS-using program and an expanded memory manager. EMS.SYS and EMS.COM give you command-line control of some of the EMS functions that are usually available only at the programming level. Since these EMS utilities are capable of granting you access to handles which may belong to other programs, you should exercise caution when using these utilities.

With the EMS programs, you can allocate and name a block of memory with the CREATE option, and optionally specify that this block of memory consists of the fastest or slowest memory on your

system. You can use the FREE option to free the expanded memory associated with a handle. You can read data from a file into expanded memory or write the data from expanded memory to a file with the LOAD or SAVE options. You can rename an EMS handle and change the amount of memory associated with it.

The most common reason for using the EMS programs is to prevent a specific application from using all of the memory in your system. By issuing an EMS CREATE command before running an application, you effectively "hide" the specified amount of memory from that application. Many programs (e.g., Microsoft Windows, AutoCAD, Quattro, Lotus 1-2-3 version 3) allocate a great deal of available memory to themselves at startup time—sometimes as much as you have on your system. By creating an EMS handle in the following fashion:

EMS CREATE handle_name 2048K

you reserve 2 megabytes of memory, identified by the name HANDLE_NAME, that other programs will see as already assigned, and therefore will not touch. Once your program has started, you could go to the DOS prompt and issue the command:

EMS FREE handle_name

to release the 2 megabytes of memory, which would leave 2 megabytes available after your application is running. Because QEMM gives out both expanded and extended memory from the same memory pool, you can use this method to withhold memory from programs that allocate their memory through EMS, XMS, VCPI, or DPMI. This method is particularly useful for preventing Microsoft Windows 3.1 standard mode from allocating all memory, so that you can run programs that get their memory through EMS, VCPI, or DPMI inside Windows.

If parts of the expanded memory in your system run at different speeds, you can use EMS to allocate memory of one speed before you load a device driver or TSR so that it can only use the faster or slower memory that remains; then you can free the memory for use by your other applications. Manifest can show you if your memory runs at different speeds.

If you are a programmer using expanded memory, you can use the LOAD and SAVE functions when you need to save and restore the contents of expanded memory during development and debugging.

You use EMS.SYS in the CONFIG.SYS file to manipulate expanded memory during the system boot sequence. You use EMS.COM in the AUTOEXEC.BAT file or directly from the DOS prompt, as needed.

To get a summary report of your expanded memory:

- **At the DOS prompt, type EMS and press Enter ↵.**

  EMS will report the total amount of expanded memory, the amount currently available and the address of the page frame.

EMS Parameters

Both EMS.SYS and EMS.COM respond to the same parameters. The parameters are described below. Some parameters have an abbreviation you can use instead of the full name; abbreviations are listed in parentheses below. Brackets ([ ]) in a statement indicate that the enclosed item is optional.

- **CREATE name amount (CR)**

  allocates expanded memory. CREATE requires two arguments: a name for the block of memory you are allocating and the amount of memory. The name may be one to eight characters long. The name need not be enclosed in quotation marks unless it contains blanks.

  You can express the amount of memory to allocate several ways: Use a number by itself to express the amount of memory in EMS pages (16K per page). Use a number directly followed by the letter K (e.g., 2048K) to express the amount in kilobytes. If you specify the number of kilobytes, the memory manager will round the number up if necessary to a multiple of 16. You can use the letter M instead of K to express a value in megabytes. You can use the argument ALL to allocate all available memory. You can use the argument ALL-nnnnnn, ALL-nnnnnnK or ALL-nnnM to allocate all available memory minus a specified number of EMS pages, kilobytes or megabytes.

  Follow the EMS CREATE command with the EMS DIR command to confirm the allocation and to determine the handle number assigned to the name.

- **CREATEFAST name amount (CFAST) and CREATESLOW name amount (CSLOW)**

  are alternate forms of the CREATE option (see above) that instruct the memory manager to allocate the memory from either faster or slower memory. Use Manifest's Expanded Memory Timings to determine if any speed difference does in fact exist.

- **DIR**

  displays a breakdown of the current expanded memory allocated. For each allocated handle, DIR gives the number of expanded memory pages associated with it, the number of kilobytes of memory those pages represent, and the name assigned to that handle, if any.

- **FREE name or number**

  frees memory and deallocates a handle. FREE requires that you specify a handle to deallocate, either by its name or number. Beware of doing this to someone else's handle.

- **HELP**

  displays help on the EMS programs and their options.

- **LOAD name or number filename**

  allows you to restore the contents of expanded memory pages that have been stored in a file. This option requires that you specify the handle name (or number) and the name of the file containing the data you want to restore. The number of pages required will be automatically allocated based on the file's size.

- **RENAME name or number new_name (REN)**

  lets you assign a new name to a handle. The first parameter to RENAME is the original handle. You may refer to this handle by its number or its name. The second argument is the new handle name. RENAME can be useful to name an unnamed handle to help you keep track of it.

- **RESIZE name or number amount (RES)**

  lets you increase or decrease the amount of memory assigned to a handle. Its two arguments are the same as those of CREATE (see above).

- **SAVE name or number filename**
  allows you to save the contents of the expanded memory pages associated with an EMS handle to a file. This option requires that you specify the handle name (or number) and the filename.

- **?**
  lists the EMS programs' parameters.

EMS2EXT.SYS

EMS2EXT.SYS converts expanded memory to extended memory, for programs that rely upon the old INT 15 method of accessing extended memory. This method is no longer widely used, and has been replaced by XMS (the Extended Memory Specification). Older versions of DOS shipped with utilities which relied upon the old INT 15 interface, most notably VDISK.SYS. These drivers have since been replaced by programs that use XMS instead, and as a result EMS2EXT is rarely useful.

EMS2EXT is not needed for programs that access memory through XMS, VCPI, or DPMI. It is intended only to provide on-the-fly control over extended memory allocated through the older INT 15 interface. Programs which support XMS, VCPI or DPMI can allocate and deallocate memory directly from QEMM's memory pool and have no need for EMS2EXT.

Even if you have an old extended memory utility, you cannot use EMS2EXT if your program expects to access extended memory directly at physical addresses above 1024K. Quarterdeck's QEXT.SYS driver, supplied with DESQview, cannot use memory supplied by EMS2EXT. Likewise, Microsoft's HIMEM.SYS cannot use memory supplied by EMS2EXT.

If you do have an old extended memory program that uses the INT 15 interface, EMS2EXT lets you allocate memory for that program out of QEMM's memory pool. The advantage of allocating this memory with EMS2EXT instead of with QEMM parameters is that the memory allocation can later be increased or decreased with the EMS.COM program without rebooting your system.

EMS2EXT is a device driver and therefore needs to be loaded with a DEVICE= statement in your CONFIG.SYS file. The statement to load EMS2EXT should look like this:

DEVICE=C:\QEMM\EMS2EXT.SYS MEMORY=nnn speed

The **nnn** parameter in **MEMORY=nnn** is the number of kilobytes of expanded memory to allocate initially (e.g., **MEMORY=512**). EMS2EXT will allocate an EMS handle named EMS2EXT for a block of memory nnnK in size. You can also load EMS2EXT without specifying any **MEMORY** parameter. EMS2EXT will be resident, but it will not allocate any memory. It will, however, reserve for itself a handle with the name EMS2EXT.

The optional SPEED parameter tells EMS2EXT to allocate faster or slower memory if there are different speeds of memory on your system. You may specify FAST, SLOW or no **SPEED** option at all.

You can, as needed, grow, or shrink the amount of extended memory for the EMS2EXT handle using EMS.COM. You can use this capability to give a program INT 15 extended memory only while it is running. For instance, if you loaded EMS2EXT with no MEMORY parameter, you could make a batch file which included the line:

EMS RESIZE EMS2EXT 128K

before running an application that needs 128K of extended memory through the old INT 15 interface. When the program terminates, another EMS statement could free the memory:

EMS RESIZE EMS2EXT 0

The memory is then returned to QEMM's memory pool for the use of other programs.

12 Utility Programs

This chapter describes several miscellaneous utility programs included with QEMM. The programs described in this chapter let you load device drivers before QEMM, detect memory on systems with more than 16 megabytes of RAM, load device drivers from the DOS prompt, ensure that Microsoft Windows runs properly with QEMM if you install Windows after QEMM, fix certain problems that occur on some Toshiba laptop PCs, fix problems that may occur if you are running LAN WorkPlace for DOS and fix bootup problems that occur with some Ultra Stor disk controllers. Read this chapter if any of these topics concern you.

DEVICE.COM: Loading Device Drivers from the DOS Prompt

DEVICE.COM is a program you can use to load certain device drivers from the DOS prompt instead of from CONFIG.SYS. DEVICE.COM will load character device drivers (e.g., a mouse driver, ANSI.SYS), but not block device drivers (e.g., drivers for disk compressors, RAM disks or CD ROM drives). In general, a block device is one that will be assigned a drive letter (e.g., E:, H:). You may want to use DEVICE.COM for the following reasons:

- To load a device driver in a DESQview or DESQview/X window, or in a Microsoft Windows DOS window. For example, if you have a program that requires ANSI.SYS, you can load ANSI.SYS in that program's window without imposing ANSI's overhead on all your other windows.
- To load a device driver from the DOS prompt when you need it.
- To load a device driver in AUTOEXEC.BAT to help QEMM's Optimize program do a more efficient job of loading programs into upper memory. Occasionally a driver in CONFIG.SYS uses enough upper memory that there is not enough left to load a subsequent driver or TSR. In this case, you can try using DEVICE.COM to load the device driver in AUTOEXEC.BAT after the later driver or TSR has been loaded. This method is especially worth trying if Optimize is unable to load a very large TSR or driver into upper memory, after loading a preceding driver into upper memory.

The syntax for DEVICE.COM is **DEVICE device_driver_pathname**

For example, to load ANSI.SYS from the DOS prompt you would type DEVICE C:\DOS\ANSI.SYS.

HOOKROM.SYS: Loading Device Drivers before QEMM

HOOKROM.SYS is a device driver that allows you to load other device drivers before QEMM in your CONFIG.SYS file. You may need HOOKROM.SYS if you need to load a device driver before QEMM386.SYS and you are using QEMM's Stealth ROM feature (i.e., you have the parameter ST:M or ST:F on the QEMM386.SYS device line in CONFIG.SYS). Though it is usually best to load device drivers after QEMM386.SYS, there are some special drivers (like the ones that manage some 80386 conversion hardware) that must load before QEMM386.SYS. These drivers may obscure information that QEMM needs to enable the Stealth ROM feature. If this is the case, QEMM386.SYS will post an error message that reads, "QEMM386: Disabling Stealth ROM because QEMM could not locate the ROM handler for INT x," where x is the number of an interrupt handler that QEMM needs to manage for the Stealth ROM process to work.

The solution to this problem is to place the line DEVICE=C:\QEMM\HOOKROM.SYS at the beginning of the CONFIG.SYS file, before the driver that needs to be loaded before QEMM386.SYS. HOOKROM will gather the necessary information for QEMM386.SYS, so that the special driver does not interfere with the Stealth ROM process.

LWPFIX: Fixing Problems with LAN WorkPlace

LWPFIX.COM is a TSR that works around problems with some versions of Novell's LAN WorkPlace for DOS. Specifically, some versions of Novell's TCPIP.EXE do not properly save and restore the state of two of the processor's 32-bit extended registers; this can cause malfunctions and crashes when other programs are using these registers. By adding the command C:\QEMM\LWPFIX.COM to your AUTOEXEC.BAT after TCPIP.EXE is loaded, you ensure that the original contents of these registers will be restored after TCPIP.EXE finishes using them.

You will need LWPFIX.COM if you are using versions 4.00 or 4.01 of LAN WorkPlace for DOS; you may need it with some later versions. LWPFIX.COM does no harm even if it is not needed, so it may be worth loading LWPFIX.COM if you are experiencing problems with any version of LAN WorkPlace for DOS. DESQview/X automatically loads a driver that performs the same function as LWPFIX.COM, so

LWPFIX.COM is only needed to fix problems that occur outside of DESQview/X.

QWINFIX: Using Microsoft Windows with QEMM

QWINFIX.COM makes Microsoft Windows 386 enhanced mode compatible with QEMM. QWINFIX does this by adding the line SystemROMBreakPoint=false to the [386Enh] section of Windows' SYSTEM.INI file. If you have Windows installed on your PC at the time you install QEMM, QEMM's installation program will run QWINFIX. If you install Windows after installing QEMM, you should run QWINFIX. To run QWINFIX:

- **Switch to Windows' directory (usually \WINDOWS).**
- **Type QWINFIX and press Enter ↵.**

QEMMREG: Displaying QEMM's Version and Serial Number

QEMMREG.COM displays QEMM's version number and your serial number. To use QEMMREG:

- **Type QEMMREG and press Enter ↵.**

SCANMEM.COM: Detecting Memory above 16 Megabytes

SCANMEM.COM is a program that scans your PC's memory, looking for memory that is not reported by the BIOS, and reports a parameter you can use to make QEMM see this memory. This program may be useful if your system has more than 16 megabytes of memory and you cannot access the memory above 16 megabytes after installing QEMM.

Some systems with more than 16 megabytes of memory do not report all of their memory through the appropriate BIOS call (the standard method for reporting how much memory is installed in a system). On such a system, QEMM will not automatically detect the memory above 16 megabytes. Certain Compaq and Dell PCs and PCs with older Micronics motherboards (e.g., some Gateways) with more than 16 megabytes of memory are the most notable examples. SCANMEM tries to locate regions of RAM that QEMM does not detect automatically when it loads. If you have a system with more than 16 megabytes of RAM and you suspect that all your memory is not available, follow the steps below:

- **First, run Manifest to see if the memory is recognized. Type MFT and press Enter ↵.**

Near the bottom of the Manifest System Overview screen, you will see a number for Total Extended Memory (pooled). If you have over 16 megabytes of RAM and the amount displayed is less than 16384K, your system's BIOS is not reporting the memory above 16 megabytes, and you should continue with the steps below.

If you are having problems accessing memory above 16 megabytes on a Dell PC, contact Dell's technical support. They may be able to supply you with an updated version of the system BIOS that fixes this problem.

You should not run the SCANMEM program when QEMM386.SYS, DOS's HIMEM or EMM386 or any other memory manager is loaded.

- **Reboot your PC without QEMM (see "Booting Without QEMM" in Chapter 2 of the *QEMM Installation Guide* for information).**
- **After rebooting, type SCANMEM and press Enter ↵.**

  SCANMEM will scan your PC's memory, and if it finds a memory region that QEMM has not detected, it will post a message listing the exact form of the USERAM=xxxxxxxx-yyyyyyyy parameter that you should put on the QEMM386.SYS device line in CONFIG.SYS. SCANMEM will list an address range in eight-digit hexadecimal format (e.g., USERAM=00100000-00206000). When you add the USERAM parameter to the QEMM386.SYS device line, use all the digits given in the address. This parameter will reclaim the memory; SCANMEM's only job is to suggest the appropriate USERAM parameter (for information on the USERAM parameter, see **Chapter 7**).

- **If SCANMEM lists the USERAM parameter, jot it down, when edit your CONFIG.SYS file and add the exact parameter SCANMEM reported to the QEMM386.SYS device driver line.**
- **Save your CONFIG.SYS file and reboot.**

  After rebooting, you should be able to access the memory above 16 megabytes. You can use Quarterdeck Manifest to verify that the memory is recognized (see the first step above).

  SCANMEM may post various messages:

  Address wrap at xxxxx, where xxxxx is a memory address, means that SCANMEM has detected that your PC's address space is smaller than the four gigabytes that the 386 processor can address. This

message is for your information and does not invalidate SCANMEM's findings.

NOUSERAM=xxxxx-yyyyy, where xxxxx and yyyyy are memory addresses, means that SCANMEM does not detect memory in the address range xxxxx-yyyyy, even though your system's BIOS has reported enough extended memory to fill these addresses. If you see this message, you may wish to use your PC's system setup to reconfigure your machine so that the BIOS reports extended memory properly.

Error: Invalid USERAM due to memory cache! means that SCANMEM has detected that the USERAM=xxxxx-yyyyy parameter that it last printed to the screen is invalid and should not be used. You should ignore only the last USERAM message printed to the screen; previous USERAM messages are valid. This error may occur if an unusual memory cache makes the contents of memory appear to be variable.

T386.EXE: Displaying the Pop-up Menu on Toshiba Laptops

T386.EXE is a program for Toshiba laptop computers which allows the Toshiba pop-up menu to appear when QEMM is enabled. T386 works on many Toshiba laptops.

If the computer is in virtual-8086 mode, Toshiba's pop-up menu will display only if the expanded memory manager calls itself "T386." The computer is always in virtual-8086 mode when expanded memory is in use or High RAM has been created. Therefore, when QEMM is performing these services you will not be able to access the pop-up menu. The T386 program makes QEMM appear to be named T386 and allows the menu to work properly. To use T386.EXE:

- **Type T386 and press Enter ↵.**

  You can load T386.EXE into upper memory by typing LOADHI T386.

  You may want to load T386 from your AUTOEXEC.BAT file, so it will run whenever you start your PC. We suggest you run Optimize after adding this or any other program to AUTOEXEC.BAT.

  To remove T386 from memory (even if it is loaded into upper memory):

- **Type T386 R and press Enter ↵.**

INT13FIX.SYS

INT13FIX.SYS prevents certain problems that can happen when the CONFIG.SYS file is being executed and problems occur on the DOS stack. INT13FIX's job is to switch away from the DOS stack and on to its own stack in conventional memory when a BIOS disk call occurs while the CONFIG.SYS file is being processed. If you give INT13FIX the /STACKSIZE=xxxx parameter, you can also change the size of INT13FIX's stack, to prevent stack overruns. The default size of the INT13FIX stack is 256 bytes; xxxx can be any value between 128 and 1024.

INT13FIX is needed with some UltraStor disk controllers to prevent "Device not found" errors during the boot process. INT13FIX with the /STACKSIZE=384 parameter also prevents "Configuration too large for memory" errors or crashes in the CONFIG.SYS file on some systems with Adaptec 1542c controllers.

If you think you need INT13FIX.SYS, load it in the CONFIG.SYS file, immediately before the QEMM386.SYS line (and after DOSDATA.SYS and any other programs loaded before QEMM386.SYS). For example:

DEVICE=C:\QEMM\INT13FIX.SYS

or

DEVICE=C:\QEMM\INT13FIX.SYS /STACKSIZE=384

Internally Used Programs

QEMM uses the following programs internally. You will not need to run these programs: TESTBIOS.COM, SWAPECHO.COM, OPT2.BAT.

13 QPI: The QEMM Programming Interface

The QEMM Programming Interface (QPI) lets programs request information or services from QEMM. Programs can use the QPI to do the following:

- Determine QEMM's status, and change that status if the system configuration allows;
- Determine QEMM's version number;
- Determine whether QEMM's Stealth ROM feature is active, and if so what Stealth ROM mode is in use;
- Determine the number of ROMs that QEMM is Stealthing and the beginning segment address and length of each ROM;
- Determine whether QEMM is supporting the system's Suspend/Resume features, and if so what interrupt these features are using;
- Determine whether QEMM is allowing or suppressing the BIOS calls that make it possible to do work while waiting for disk activity to complete, and tell QEMM to allow or suppress these calls if the system configuration allows;
- Copy all or part of the contents of a Stealthed ROM into a buffer;
- Determine the physical memory mapped to any linear memory address, and change the page table so that any page of physical memory is mapped to any linear memory address;
- Read or write I/O ports, even if QEMM is trapping access to those ports;
- Determine whether QEMM is trapping access to I/O ports, and tell QEMM to trap access to ports on the calling program's behalf;
- Install a software routine that performs whatever actions the program requires when a given I/O port is accessed;
- Simulate a hardware interrupt in such a way that it goes to the correct DESQview or DESQview/X window.

You can find sample QPI code and programs on the Quarterdeck bulletin board and other electronic support locations.

Getting the QPI Entry Point

The first step in using the QPI is getting the double-word address of the QPI entry point.

The method of obtaining the entry point address that is described here is available only in versions of QEMM higher than 6.00. Programs that want to run with QEMM 5 must use a less straightforward INT 2F interface to get the QPI entry point; for more information, contact Quarterdeck Developer Support and ask for the QDMEM interface document.

QEMM defines a DOS device driver called QEMM386$. To obtain the entry point for QPI, do an IOCTL Read Control String call (INT 21, function 4402h) to read four bytes from this device driver. Here is a code sequence that demonstrates the details:

```
QEMMDeviceName db 'QEMM386$',0
QPIEntryPoint  dd ?

GetQPIEntryPoint proc
          mov  dx,offset QEMMDeviceName
          mov  ax,3d00h
          int  21h                          ; Try to open QEMM386$
          jc   NoQEMM                       ; If CY, QEMM not present
          mov  bx,ax                        ; Save file handle in BX
          mov  dx,offset QPIEntryPoint      ; Store the entry point here
          mov  cx,4                         ; Set up to read 4 bytes
          mov  ax,4402h                     ; IOCTL Read Control String
          int  21h
          pushf                             ; Save the error code
          mov  ah,3eh                       ; Close the handle
          int  21h
          popf                              ; Restore the error code
          jc   NoQEMM                       ; If CY, QEMM is pre-6.00
          ret
NoQEMM:   stc
          ret
GetQPIEntryPoint endp
```

QPI Functions

Once you have stored the address of the QPI entry point, you make all calls to QPI by loading AH or AX with the function number of the call, setting the other registers to values appropriate to the function, and making a far call to the entry point. The carry flag is set on return if there is an error, or if the function number is not valid in that version of QEMM.

The QPI calls of interest to third-party programmers are listed below. The version in which each of the calls was implemented is noted.

The QPI_GetStatus Call

The QPI_GetStatus call tells you whether QEMM is on or off, and whether it is in auto mode (see the AUTO/ON/OFF parameter in **Chapter 7** for more information). All versions of QEMM support this call.

```
QPI_GetStatus                   EQU     0
      ; Takes       AH = 0
      ; Returns     AL = 0 if on
      ;             AL = 1 if auto/on
      ;             AL = 2 if off
      ;             AL = 3 if auto/off
```

The QPI_SetStatus Call

The QPI_SetStatus call lets you set the status of QEMM. If QEMM is forced on by a parameter (like the RAM parameter) or other services that it provides, this call will have no effect. You should therefore make the QPI_GetStatus call after the QPI_SetStatus call to see if the first call was successful. All versions of QEMM support this call.

```
QPI_SetStatus                   EQU     1
      ; Takes       AH = 1
      ;             AL = 0 if on
      ;             AL = 1 if auto/on
      ;             AL = 2 if off
      ;             AL = 3 if auto/off
```

The QPI_GetVersion Call

The QPI_GetVersion call returns the QEMM version number in Binary Coded Decimal form in AX and BX. For instance, the call will return BX = 0750 (not BX = 0732) for QEMM version 7.5. All versions of QEMM support this call.

```
QPI_GetVersion                  EQU     3
      ; Takes       AH = 3
      ; Returns     BH = major version (in Binary Coded Decimal)
      ;             BL = minor version (in Binary Coded Decimal)
      ;             AX = same as BX
```

The QPI_GetInfo Call

In QEMM version 6.00 and later, the QPI_GetInfo call returns an ASCII letter in CL that tells QEMM's Stealth ROM mode (if any), and a number in CH that tells which interrupt (not IRQ) QEMM is monitoring (if any) to support Suspend/Resume features. In QEMM version 7.00 and later, the call also returns the size of QEMM's disk buffer in DL and a bit map of information about the disk buffer in BH. Bit 1 of BH will be on if the disk buffer has already been used; bit 0 will be on if QEMM is buffering only INT 13s into the page frame (DISKBUFFRAME) and off if all INT 13s into nonlinear memory are

being buffered (DISKBUF). Note that other registers are not preserved by this call.

```
QPI_GetInfo                    EQU    1E00h
      ; Takes       AX = 1E00
      ; Returns     BH = xxxxxxAB
      ;                    where A = 1 if disk buffer has been used
      ;                              yet, 0 if not
      ;                          B = 1 if DISKBUFFRAME buffer, 0 if
      ;                             DISKBUF buffer - not valid if
      ;                             DL = 0
      ;             BL = reserved
      ;             CL = Stealth ROM type (0 for no Stealth ROM,
      ;                    "M" or "F" otherwise, other
      ;                    Stealth ROM types possible
      ;                          in future)
      ;             CH = Suspend/Resume INT number (0 = none)
      ;             DL = size of QEMM disk buffer in K
      ;                    (if 0, disk buffer doesn't exist)
      ;           DH, DI, SI = reserved
      ;
```

The QPI_GetStealthCount Call

The QPI_GetStealthCount tells how many ROMs QEMM is Stealthing. QEMM versions 6.00 and later support this call.

```
QPI_GetStealthCount          equ  1E01h
      ; Takes       AX = 1E01
      ; Returns     BX = number of ROMs that are Stealthed
```

The QPI_GetStealthList Call

The QPI_GetStealthList call gives the same information as the QPI_GetStealthCount call, and also fills a buffer with information on the location and size of each Stealthed ROM. QEMM versions 6.00 and later support this call.

```
QPI_GetStealthList           equ 1E02h
      ; Takes       AX = 1E02
      ;             ES:DI= buffer to hold the list of Stealthed ROMs
      ; Returns     BX = number of ROMs that are Stealthed
      ;             Table at ES:DI will be filled in with:
      ;                 dw ROM start segment
      ;                 dw Length of ROM in paragraphs
      ;             for each ROM that is Stealthed
```

The QPI_GetPTE Call

The QPI_GetPTE call returns the page table entry for any logical page in the first 1088K of memory. In other words, if you pass this call the address of any page in the first 1088K of memory, the call will return (in an extended register) the doubleword page table entry for that address, which includes, among other things, information about which physical page of memory QEMM has mapped to the logical address that you provided. QEMM versions 6.00 and later support this call.

CX should contain the number of the logical page in memory that you want to affect; the highest valid CX is 010F. (Page numbers refer to consecutive 4K sections of memory, aligned on 4K boundaries: that is, 0000 refers to paragraph 0000-00FF, 0001 to 0100-01FF, etc.) EDX (the extended DX register) should contain a page table entry, in the following format:

- Bit 0 is the Present bit. Any access to a page with this bit off causes a page fault.
- Bit 1 is the Read/Write bit. Any writing to a page with this bit off causes a page fault.
- Bit 2 is the User/Supervisor bit. Any access to this page when the processor is at privilege level 3 causes a page fault.
- Bits 3 and 4 must be 0.
- Bit 5 is the Accessed bit. Any read or write of the page causes the processor to turn on this bit.
- Bit 6 is the Dirty bit. Any write to the page causes the processor to turn on this bit.
- Bits 7 and 8 must be 0.
- Bits 9, 10, and 11 are available for systems programmer use.
- Bits 12 through 31 are the page number.

For instance, a page table entry 000FF007 means physical page number 000FF (paragraph FF00-FFFF), which is not yet accessed nor dirty, but which is present, writable and user-accessible.

```
QPI_GetPTE                  equ 1F00h
        ; Takes     AX = 1F00
        ;           CX = page number
        ; Returns   EDX = page table entry for that page number
```

The QPI_SetPTE Call

The QPI_SetPTE call lets you set the page table entry for any logical page in the first 1088K of memory. In particular, this means that you can tell QEMM to map any 4K of memory to any 4K-aligned address below the 1088K mark. QEMM versions 6.00 and later support this call. See the section above on the QPI_GetPTE call for more information.

```
QPI_SetPTE                      equ     1F01h
        ; Takes         AX = 1F01
        ;                CX = page number
        ;                EDX = page table entry to set at that page number
```

The QPI_GetVHIInfo Call

The QPI_GetVHIInfo call, in conjunction with the QPI_SetVHIInfo call, is primarily used by disk cache developers who wish to get information on QEMM's safety precaution of suppressing the BIOS INT 15 function 90 callout. This safety precaution, known as VirtualHDIRQ or VHI (see the section on the VIRTUALHDIRQ:N parameter in **Chapter 7** for more information), is normally in effect only when the disk interrupt INT 13 is being Stealthed. If you have verified that the use of INT 15 fn 90 in the disk cache you are developing is compatible with QEMM's Stealth ROM feature, you will want to tell QEMM to allow INT 15 function 90.

The call returns flags in BL that give the VHI state. Bit 7 will be on whenever QEMM is Stealthing INT 13 (if INT 13 is not Stealthed, QEMM never suppresses INT 15 function 90); bit 0 will be on if QEMM is currently suppressing INT 15 function 90. Bits 1-6 of the VHI bit map are reserved. QEMM versions 6.00 and later support this call.

```
QPI_GetVHIInfo                  equ     2000h
        ; Takes         AX = 2000
        ; Returns       BL = AxxxxxxxB
        ;                where A = 1 if VHI is being paid attention to
        ;                     B = 1 if VHI is currently enabled
        ;                 (i.e. INT 15 fn 90 is currently suppressed)
        ;                     x = reserved
```

The QPI_SetVHIInfo Call

The QPI_SetVHIInfo call lets you turn on or off QEMM's safety precaution of suppressing INT 15 function 90 whenever the disk interrupt INT 13 is being Stealthed. (See the section above on the QPI_GetVHIInfo call for more information.) To request a VHI state, set BL to 1 to suppress INT 15 function 90, or set BL to 0 to allow INT 15 fn 90. QEMM will return the previous VHI flags in BL. If bit 7 of the returned flags is off, then QEMM is not paying attention to the VHI state, and your request did not have an effect. QEMM versions 6.00 and later support this call.

```
QPI_SetVHIInfo      equ     2001h
        ; Takes         AX = 2001
        ;                BL = xxxxxxxxB that you want (bit 7 is ignored)
        ; Returns       BL = AxxxxxxxB of previous VHI state;
        ;                if A of output = 0, B of input was ignored
```

The QPI_CopyStealthRoms Call

The QPI_CopyStealthRoms call tells QEMM to copy the contents of part or all of a Stealthed ROM into a buffer in conventional memory. This is the only reliable to access the contents of a Stealthed ROM. QEMM versions 6.00 and later support this call.

```
QPI_CopyStealthRoms         equ     2100h
        ; Takes       AX = 2100
        ;             DS:SI = Original address of ROM to copy
        ;             ES:DI = Destination address in conventional memory
        ;             ECX = # of bytes to copy
        ; Returns     CY if no stealth or if DS:SI not within C000-FFFF
```

I/O Trapping

The following calls make up the Quarterdeck QEMM I/O Trapping Programming Interface. This interface allows a real-mode program to specify I/O ports that QEMM should trap access to, as well as I/O callback routines that QEMM will call whenever one of these I/O ports is accessed. Using this interface, you can emulate hardware devices that are accessible via I/O ports.

When QEMM traps an I/O port, all accesses of that port, whether input or output, are intercepted by QEMM. (QEMM traps certain I/O ports itself, for proper management of virtual-8086 mode.) Whenever an I/O port that a program has asked QEMM to trap is accessed, QEMM calls a real-mode I/O callback routine. The same callback routine is called for all trapped I/O ports.

A program that wishes to trap an I/O port should:

1) Use the QPI_GetVersion call to make sure that the version of QEMM is 7.03 or later. Earlier versions do not support most of the I/O Trapping Programming Interface. Alternatively, if the version of QEMM does not support the call you have made, the call will return with the carry flag set;

2) Issue a QPI_GetPortTrap call to determine that another program is not already trapping the port. If another program is trapping the port, it is generally advisable not to install your port trap;

3) Get the address of the existing callback routine with the QPI_GetIOCallback call. Because there is only one I/O callback routine, and because multiple programs may request I/O trapping, your callback routine must jump to the previous callback routine whenever your routine is not interested in the I/O port being accessed;

4) Install its own far routine as the new callback routine, using the QPI_SetIOCallback call. Your callback routine will be passed the following information:

```
AX = Data for output
CX = Type of I/O (see flag bits defined below)
DX = Port number
IF = 0 (interrupts are disabled)
```

When the callback routine has finished its work, it should return far with all registers other than CX and DX preserved. If the routine is called to get input from a port, AX should be modified.

The bit-mapped word in CX contains the following information:

```
IOT_Output          equ     0000000000000100b
        ;           bit 2 is 1 if output,
        ;                   0 if input



IOT_Word            equ     0000000000001000b
        ;           bit 3 is 1 if word I/O,
        ;                   0 if byte I/O



IOT_IF              equ     0000001000000000b
        ;           bit 9 is the same as the
        ;               caller's interrupt flag
```

5) Specify which port to trap with the QPI_SetPortTrap call.

If your program does not stay resident forever, it should do the following before exiting:

6) Use the QPI_GetIOCallback call to make sure that no one has installed a callback routine after yours. If a handler is installed after yours, you should remain resident;

7) Remove its trap with the QPI_ClearPortTrap call. This call will clear all traps on a particular I/O port, unless QEMM is trapping that port for itself, in which case QEMM's trapping alone will remain in effect for that port;

8) Remove its callback by using the QPI_SetIOCallback call to set the previously existing callback routine.

The following restrictions apply to this interface:

- Only INs and OUTs of words or bytes are supported, not INs and OUTs of doublewords. Also, string I/O (the INS and OUTS instructions) are supported only as of QEMM version 7.5.
- QEMM cannot trap the I/O of VCPI protected-mode programs. Furthermore, because Quarterdeck's DPMI driver (QDPMI) is implemented as a VCPI client, QEMM cannot trap the I/O of DPMI clients either. Furthermore, these traps are no longer in effect when Microsoft Windows 386 enhanced mode is running.

The following four calls allow you to bypass port trapping and read and write I/O ports directly. QEMM 5.00 and later versions support these four calls.

```
QPI_UntrappedIORead equ     1A00h
      ; Takes       AX = 1A00
      ;             DX = port to read
      ; Returns     BL = value read



QPI_UntrappedIOWrite equ    1A01h
      ; Takes       AX = 1A01
      ;             DX = port to write
      ;             BL = value to write



QPI_UntrappedIOReadIndexed   equ    1A02h
      ; Takes       AX = 1A02
      ;             DX = base port to read
      ;             BH = index into base port
      ; Returns     BL = value read



QPI_UntrappedIOWriteIndexed   equ    1A03h
      ; Takes       AX = 1A03
      ;             DX = base port to write
      ;             BH = index into base port
      ;             BL = value to write
```

The QPI_UntrappedIO Call

The QPI_UntrappedIO call performs the same functions as the QPI_UntrappedIORead and the QPI_UntrappedIOWrite calls, but it uses register values that are similar to the ones QEMM passes to your I/O callback routine (including the flags in CX that give information about the type and size of the I/O and the caller's interrupt flag). This call may therefore be easier to use from within an I/O callback routine. QEMM 7.03 and later support this call.

```
QPI_UntrappedIO         equ     1A04h
        ; Takes       AX = 1A04
        ;             BX = value to write
        ;             DX = port to read or write
        ;             CX = type of I/O (see description above of CX passed
        ;             to callback routine)
        ; Returns     BX = value read
```

The following five calls are described in the introduction above. QEMM 7.03 and later versions support these calls.

```
QPI_GetIOCallback       equ     1A06h
        ; Takes       AX = 1A06
        ; Returns     ES:DI = previous I/O callback function


QPI_SetIOCallback       equ     1A07h
        ; Takes       AX = 1A07
        ;             ES:DI = new I/O callback function


QPI_GetPortTrap         equ     1A08h
        ; Takes       AX = 1A08
        ;             DX = I/O port number
        ; Returns     BL = 0 if port not trapped, BL = 1 if port already trapped


QPI_SetPortTrap         equ     1A09h
        ; Takes       AX = 1A09
        ;             DX = I/O port number


QPI_ClearPortTrap       equ     1A0Ah
        ; Takes       AX = 1A0A
        ;             DX = I/O port number
```

The QPI_SimulateHWInt Call

The QPI_SimulateHWInt call can be used by callback routines that wish to simulate a hardware interrupt. When DESQview or DESQview/X is running, the interrupt handler that should receive the interrupt may be in a different process from the current one. Use QPI_SimulateHWInt to simulate an interrupt properly when DESQview or DESQview/X is running. *The DESQview API Reference Manual* describes how to determine when DESQview is running. QEMM 7.03 and later versions support this call.

```
QPI_SimulateHWInt       equ     1C04h
        ; Takes         AX = 1C04
        ;               BX = interrupt number to generate
```

NOTES:

Appendices

A Troubleshooting

This section contains tips on handling some common problems users may experience after installing QEMM. If your question is not answered in this section, you will find a list of "README" files that contain further troubleshooting tips in **Appendix B**.

Some of the troubleshooting instructions will ask you to edit your CONFIG.SYS file. If you edit that file be sure to use a text editor (e.g., EDIT supplied with DOS 5 and 6). If you use a word processor, be sure it is in ASCII mode (see the word processor's manual for details).

Alternatively, you can use QEMM's Setup program. Just type **QSETUP** and press **Enter** ↵, press **Enter** ↵ again to go to the QEMM Setup menu. You can manually edit CONFIG.SYS by selecting **Edit the proposed CONFIG.SYS**. Or you can have the Setup program add or remove several of QEMM's most important parameters by selecting **Review or change QEMM parameters** and following the on-screen instructions.

Your PC or a particular software package does not function properly with QEMM.

QEMM includes a text file that contains information about running QEMM with particular hardware or software. To see the information, use a text editor to view the file C:\TECHNOTE\PRODUCTS.TEC. You can also use a word processor running in ASCII mode or QEMM's Setup program to view the file. To view the file with QEMM Setup:

- Type **QSETUP** and press **Enter** ↵.
- Press **Enter** ↵ again to get to the QEMM Setup menu.
- Type **H** to select **View QEMM hints, technotes and READ.ME.**
- Type **T** to view technotes.
- When the list of technotes displays, press **Page Down** and select **Solutions for Problems with Specific Products.**

The products technote will display, and you can use PageDown and PageUp to scroll through it.

Your PC does not function properly after installing QEMM.

1. Follow the steps below to boot your PC without QEMM.

- Reset your system. Use the power switch if necessary.
- Wait until you hear a beep, then hold down the Alt key until the boot sequence stops.
- If you are using QEMM's DOS-Up feature, you will see a message asking if

you want to unload DOSDATA; press Esc to unload DOSDATA, then hold down Alt again.

You will see the following message: "QEMM: Press ESC to unload QEMM or any other key to continue with QEMM."

- **If your system does not beep on bootup, hold down Alt for three or four seconds after you reset the system.**
- **Press the Esc key.**

  Your system will then proceed with the boot sequence. QEMM will not be loaded and no programs will be loaded into High RAM. If your problem still occurs, it is not related to QEMM386.SYS; try removing other QEMM programs or investigating other software and hardware. If the problem no longer occurs, you can now modify the QEMM386.SYS driver line in your CONFIG.SYS file (as explained below) to investigate the problem.

2. Next you should disable some of QEMM's optional features to determine whether they may be causing problems on your system. To do this:

- **Run QEMM's Setup program and disable DOS-Up, the DPMI Host and Stealth D*Space (for information on QEMM Setup, see Chapter 2).**

  Note that you should choose "No", and not "Partial" in response to the DOS-Up option. Note also that if you are not using DOS 6's DoubleSpace or DriveSpace, the option to enable or disable Stealth D*Space will not appear as a Setup option.

- **Once you have disabled these features select Save Configuration and Quit. If you are in Microsoft Windows or DESQview, exit to DOS. Reboot your machine.**

  If your problem persists, but was solved by disabling QEMM in step 1 above, the problem is likely related to the QEMM386.SYS driver and you should proceed to step 3 below.

  If your problem is now solved, one of the features you have just disabled is likely in conflict with some other aspect of your system. Use QEMM Setup to re-enable each feature, one at a time, save your configuration and reboot, until you determine which feature seems to be causing the conflict. When you find the feature that is causing the problem, you may want to contact Quarterdeck Technical Support for assistance in getting the feature to work properly on your system. Be sure to run Optimize after you have determined your final configuration (for information on Optimize, see **Chapter 3**).

3. You may have a Stealth-related problem. Edit your CONFIG.SYS file and see if one of the Stealth ROM parameters (ST:F or ST:M) is on the QEMM386.SYS line; if there is no such parameter, skip to step 4 now.

   Make a note of which parameter (ST:F or ST:M) is there, then delete that parameter. Save CONFIG.SYS and reboot. If your PC functions properly, the problem is related to Stealth ROM; run the Analysis procedure described on *page 125*. If your PC does not function properly, continue with step 4, rebooting without QEMM (see step 1 above) if you cannot edit CONFIG.SYS otherwise.

4. Edit CONFIG.SYS, and if there are any INCLUDE (I) or ROM parameters on the QEMM386.SYS line, remove them and reboot. If your system functions properly now, you may want to try to narrow down which parameter or region caused the problem by replacing the parameters one by one.

5. There may be a conflict between QEMM and one of your PC's devices (e.g., a network adapter) or features. If you have the RAM parameter on the QEMM386.SYS line in CONFIG.SYS, remove the RAM parameter, save CONFIG.SYS then reboot. If you do not have the RAM parameter, skip to step 6.

   If your PC now functions properly, a device may have been using some area between 640K and 1024K in such a way that QEMM could not detect it. You should exclude that area from QEMM's management. To do that, run the Analysis procedure and add the necessary EXCLUDE parameters to the QEMM386.SYS line (see ***page 125***). After following the recommendations of the Analysis report and running Optimize, see if the problem is corrected.

6. If you still have not determined the conflict, then you should create a "pure environment" to isolate the source of the trouble. To create a pure environment, you need to temporarily remove all lines that are not absolutely necessary from your AUTOEXEC.BAT and CONFIG.SYS files.

   a. First, make backup copies of these files (just in case you need them later) by copying them to CONFIG.XXX and AUTOEXEC.XXX.

   b. Next, edit CONFIG.SYS and AUTOEXEC.BAT with a text editor (e.g, QEMM Setup's editor, DOS EDIT or EDLIN) and "comment out" all lines except those that are absolutely necessary. To comment out a line, add the word REM at the beginning of the line (REM causes the line to be ignored when the file is processed). When you edit CONFIG.SYS, do *not* comment out the QEMM386.SYS device driver line, the FILES line, or any drivers that are absolutely necessary (e.g., a hard disk manager). Also, you need not comment out the PROMPT line in AUTOEXEC.BAT. If you are troubleshooting problems with a specific device, do not comment out the lines related to that device (e.g., if your network is not functioning properly, do not comment out the network drivers or any lines relating to the network). Once you have edited each file, save it.

   c. Reboot your PC and test to see if the problem still occurs. If it does, you may have a hardware problem or your software may be incompatible with your hardware. You may need to have your hardware checked out. If you think the hardware is OK, you may want to contact the software publisher to see if they are aware of a solution. If you do not suspect a hardware or software problem, skip to step 7.

   If the problem does not occur, one or more of the lines you commented out may have been causing the problem. Examine CONFIG.SYS and AUTOEXEC.BAT files to see if you can determine which commented line may be at fault.

d. Remove the word REM a line at a time, reboot and test to see if the problem still occurs. When you are able to recreate the problem, you have isolated the offending line. You have a few alternatives here: 1) contact the publisher of the driver or TSR and see if they know of a solution, 2) remove the driver or TSR from your CONFIG.SYS or AUTOEXEC.BAT file, or 3) contact Quarterdeck Technical Support for further assistance. If you are able to fix the problem and made the necessary changes to your CONFIG.SYS file and AUTOEXEC.BAT files, run Optimize (see **Chapter 3**).

   If you have not solved the problem, continue with step 7.

7. If you are NOT using the RAM parameter (or you were using it and removing it did not help) and the computer will still not boot properly, there is some other conflict. Try adding the following group of parameters (listed here in their abbreviated form) to the QEMM386.SYS device driver line in your CONFIG.SYS file: DB=2, CF:N, FILL:N, MR:N, RH:N, SH:NONE, TR:N, TM:N, XBDA:N. For information on these parameters, see **Chapter 7**.

   If adding the above parameters fixes the problem, remove them one at a time and reboot until the problem returns. At that point you have found at least one parameter that helps fix the problem, so add it back. Repeat this process for the remaining parameters. You may want to read about these parameters to see what they do and to find out the ramifications of using them (see **Chapter 7**).

8. If you still have not solved the problem, try reinstalling QEMM. If that does not help, contact Quarterdeck Technical Support.

Optimize does not complete successfully or your system does not initialize properly after running Optimize.

QEMM may be using certain upper memory addresses needed by your system or by a program. In this case, you can rerun Optimize and have it attempt to uncover RAM areas that should not be used.

- **Switch to the QEMM directory by typing CD \QEMM and pressing Enter ↵.**
- **Type OPTIMIZE /AUTO and press Enter ↵.**

This will begin the Optimize /AUTOEXCLUDE detection. You will see a message asking you to power off your PC, then turn it back on. Do not do a warm reboot or use your PC's Reset button.

- **Power off your PC, wait five seconds, then turn it back on.**

Optimize may reboot your system again. When your PC finishes booting, you will see Optimize's Exclusion Processing screen, listing in hexadecimal notation the upper memory addresses Optimize thinks should be excluded from use as High RAM.

If there are several addresses listed, you may not need to exclude them all. If you feel comfortable doing so, we suggest you view the exclusions and try to determine which ones are necessary. If you want to accept Optimize's recommendations, just press **Enter** ↵ to continue. Otherwise, to view the exclusions:

- **Press F1.**

You will see a list of programs and drivers. Some of them are followed by addresses—these are the upper memory addresses that the program or driver accesses. If there are multiple address ranges, you may not need to exclude them all. We suggest you first try excluding only those that are related to programs or drivers associated with an adapter card (e.g., a network card). You can always run OPTIMIZE /AUTO later if you need to exclude more. To remove an item from the exclusion list:

- **Use the arrow keys to highlight the Y in the item's first column and press the spacebar to change it to N.**

If there is a down arrow at the bottom of the list, there are more items below. You can press Page Down to see more items.

- **When you are finished editing the exclusion list, press Esc.**

You are back at the Exclusion Processing screen. To continue optimizing:

- **Press Enter ↵ and follow the on-screen instructions.**

Optimize will add EXCLUDE (X) parameters to the QEMM386.SYS device line in your CONFIG.SYS. For information on the EXCLUDE parameter, see **Chapter 7**.

If there are still problems after running OPTIMIZE /AUTO, you may want to run it again and exclude any addresses you may have removed from the exclusion list. If the Optimize procedure still does not complete, perform the steps in the first troubleshooting tip in this section.

When running Optimize, your system locks up before Optimize is able to reboot.

You can solve this problem by starting Optimize with the /NOARAM or the /NOFLUSH switches. See **Chapter 3** for more information on these parameters.

Your system freezes on bootup, just after QEMM's DOSDATA message displays.

If your system has been functioning properly with QEMM and you begin to experience this problem, we suggest you use a virus checker program to check your PC for viruses.

When trying to load a TSR or device driver high, you see the message, "Not enough room to load high."

Try rerunning Optimize (see **Chapter 3**).

After optimizing, a TSR or device driver does not load high.

Many TSRs and device drivers require more memory to initialize than they do once they are resident. For example, a 10K mouse driver may require 22K to initialize. So, if the largest single High RAM region were 20K, QEMM may not be able to load the driver high. We suggest you find out how much memory the TSR or device driver that did not load high needs to initialize. Run Optimize and use the "Modify Data" option (available when you see the "Analysis Complete" screen). The number of K that your program needs to initialize is listed next to your program's name in the Initial Size column.

Once you find out how much memory is needed, escape out of the "Modify Data" screen and use the "Region Layout" option to see the sizes of your system's available

High RAM areas. What you do next depends on whether any of the High RAM areas are large enough to load your TSR or device driver.

Microsoft CD ROM extensions software (MSCDEX) does not load high after running Optimize.

When loaded in its default state, the MSCDEX driver software may occupy a larger chunk of memory than is available on your system. You can substantially reduce the resident size of MSCDEX by specifying the /E parameter on the MSCDEX command line in your AUTOEXEC.BAT file. The /E tells MSCDEX to load part of its code into expanded memory, which will reduce its overhead in the first megabyte of memory. Once you have added the /E parameter to MSCDEX, re-run Optimize and see if MSCDEX will load high (see **Chapter 3** for information on running Optimize).

If you have a High RAM area large enough to load the item high: You may be able to create enough room to load the item high by changing the load order of device drivers or TSRs. Run Optimize and use the "What if" option; try to load the bigger items first (see **Chapter 3**).

If you do not have a High RAM area large enough to load the item high: You may have an adapter that uses upper memory addresses and the adapter RAM or ROM is splitting up a large upper memory region. Use QEMM.COM's Type Map report to find out (see **Chapter 9**). If this is the case, you may be able to move the adapter's RAM or ROM to another upper memory area, thereby making enough contiguous space to load the TSR or device driver high. See the adapter's manual for information on using an alternative address for the adapter's RAM or ROM. Also, if QEMM's Stealth ROM feature is not enabled, try enabling it to get more High RAM (see **Chapter 5**).

One of your TSRs or device drivers does not work well when loaded high, but Optimize still tries to load it high when you re-Optimize your system.

You can create a file called OPTIMIZE.NOT in your QEMM directory that will prevent some of your programs from being loaded high by Optimize. See "Excluding TSRs and Device Drivers from Optimize" in **Chapter 3** for details on how to create the OPTIMIZE.NOT file.

Solving a memory conflict with the Analysis procedure.

An Analysis procedure determines what upper memory addresses are accessed by programs and hardware devices. It will recommend addresses that can or should not be used as High RAM or for EMS mapping. You may want to run an Analysis if:

- After installing QEMM, a program functions or displays improperly or crashes the system.
- A hardware device (e.g., floppy disk, CD ROM drive) no longer works properly.
- You want to see if it is possible to create more High RAM to load device drivers or TSRs that QEMM was unable to load high.

1. The first step is to determine if QEMM's Stealth ROM feature is enabled and, if so, which Stealth ROM method, F or M is

in use. At the DOS prompt, type QEMM and press Enter ↵.

A report summarizing QEMM's status will display. If you see the line Stealth ROM Type = M or Stealth ROM Type = F, Stealth ROM is enabled; remember which letter is listed, M or F.

2. Use a text editor to edit your CONFIG.SYS file and type REM followed by a space at the beginning of the line that starts as follows:

   DEVICE=C:\QEMM\QEMM386.SYS (REM causes the line to be ignored).

   The line should look something like this:

   REM DEVICE=C:\QEMM\QEMM386.SYS

3. Add a new line directly below the line you just edited. What the line says depends on whether you have Stealth ROM enabled.

   If you are using Stealth ROM: Add the following line, substituting the appropriate Stealth letter, M or F for the x in ST:x.

   DEVICE=C:\QEMM\QEMM386.SYS ON MAPS=0 ST:x

   Be sure to type the above text on a single line.

   If you are not using Stealth ROM: Add the following on a single line:

   DEVICE=C:\QEMM\QEMM386.SYS ON MAPS=0

4. Save your CONFIG.SYS file and reboot your PC.

5. What you do next depends on what kind of analysis you want to do.

Partial Analysis: If you are troubleshooting a program that does not display or run properly, or a hardware operation that does not work properly (e.g., a floppy drive), you will want to do a partial analysis, which simply entails using the program or device that is giving you trouble since you installed QEMM.

To do a partial analysis, just run the programs that have not been working and access their basic features. Or, if you are having trouble with a particular hardware operation, perform that operation. Things should work properly now; if there are still problems, restore the original QEMM386.SYS line in CONFIG.SYS and try some of the procedures explained on *page 167*. When you are done, skip to step 6.

Comprehensive Analysis: If you want to fully analyze your system and programs to see if you can gain additional High RAM, you will want to do a comprehensive analysis. This kind of analysis is somewhat time-consuming—you will need to run all your programs and access all hardware devices.

To do a comprehensive analysis, run your programs and use their basic functions (be sure to access any graphics-based programs). If you use DESQview, run its programs and features, then quit. Do not use any hardware or memory analysis utilities other than QEMM. Access all your PC's hardware and peripherals. If you have two monitors, display information on both of them. Access all disk drives. Format a diskette. Print a document. If you have a network adapter card or

terminal emulation adapter, access the network or terminal emulator. When you are done, continue with step 6.

Be aware that the new QEMM line you added for the Analysis procedure does not include the RAM parameter. That means you no longer have High RAM, so device drivers and TSRs will load low and you will not have as much conventional memory to run programs. We will add the RAM parameter back later.

Special note to DESQview and DESQview/X users: DESQview and DESQview/X use available portions of upper memory for themselves. If you run an Analysis procedure to find out what upper memory areas a program accesses when running under DESQview or DESQview/X, the results of the Analysis procedure will be misleading—DESQview's use of upper memory will prevent you from seeing accesses by the program in question. You can get around this by running DESQview or DESQview/X with a parameter that prevents DESQview itself from using upper memory. To do that, start DESQview by typing DV /X:A000-FFFF (or DVX /X:A000-FFFF for DESQview/X). Be aware that DESQview will be using more conventional memory since you are preventing it from using upper memory. That means there will be less memory for each window. If your program requires too much memory to fit in the smaller window, you cannot run the program under DESQview, so the Analysis will not help. If this is the case and you are troubleshooting a program, try changing the program's protection level in the program's DVP. See your DESQview or DESQview/X manual for information on changing a DVP.

6. Display the Analysis report by typing **QEMM ANALYSIS** or **QEMM ANALYSIS MAP**. If you are unfamiliar with hexadecimal addressing, we suggest using QEMM ANALYSIS without MAP—the report will be easier to interpret.

   The default view is in list format which explicitly lists addresses you may instruct QEMM386.SYS to include or exclude.

   This report uses three terms:

   OK (O): Memory areas QEMM has properly identified.

   Exclude (X): Areas of memory that have been accessed, but that QEMM expected would remain unaccessed. You should use the QEMM386.SYS's EXCLUDE parameter to prevent QEMM from using these areas.

   Include (I): Upper memory areas that are reserved by QEMM but have not been used by a program. You may use QEMM386.SYS's INCLUDE parameter to allow QEMM to use these areas.

7. Examine the Analysis report and jot down the suggestions. Edit CONFIG.SYS and delete the line you added earlier. It looks like one of the following:

   DEVICE=C:\QEMM\QEMM386.SYS ON MAPS=0 ST:x

   or

   DEVICE=C:\QEMM\QEMM386.SYS ON MAPS=0

8. Delete the word REM you added at the beginning of the original QEMM386.SYS line. Then, use the EXCLUDE or

INCLUDE parameters to exclude or include any memory addresses specified in the Analysis report. You can abbreviate EXCLUDE as X, and INCLUDE as I. For example, if the Analysis report recommended excluding B000-B7FF, you would add EXCLUDE=B000-B7FF to the end of the QEMM386.SYS line. For information on the INCLUDE and EXCLUDE parameters, see **Chapter 7**.

9. Save the CONFIG.SYS file, reboot your computer, and rerun Optimize (see **Chapter 3** for information on Optimize).

If you see an Exception 6 or Exception 13 message...

An Exception 6 or 13 message is an error message from your PC's main processor, giving information about an invalid instruction. The invalid instruction may have come from a bug in a program, or may result from a conflict between two programs or between a program and a hardware device.

Exception errors are not QEMM errors, but rather errors from your PC's main processor that QEMM passes on to you as a courtesy, so you will know what happened and can take steps to deal with them. Without QEMM, your system may have crashed or hung without warning.

A program that uses protected mode but does not support VCPI (Virtual Control Program Interface) or DPMI (DOS Protected Mode Interface) cannot run under QEMM. We suggest you contact the developer to see whether they provide support for these specifications.

Here is a tip for solving the problem that caused the Exception error: Look at the Exception 6 or 13 message and write down the address directly following the words "Exception 6" or "Exception 13" (e.g., C801:012F). This is the memory address of the code being executed when the exception occurred. (The other numbers are the contents of the CPU's registers at the time, and fifteen bytes of the code at the exception location.)

- If the address you wrote down starts with a letter from A to F, the address is in upper memory. If so, run Manifest and look at the First Meg Programs display to see if any program's Memory Area contains the first four digits of the address you wrote down. If so, try loading the program into conventional memory instead. To do that, delete the LOADHI syntax from the line in CONFIG.SYS or AUTOEXEC.BAT that loads the program (see **Chapter 8** for information on LOADHI).

- If the address you wrote down begins with a number or if you have not solved the problem, follow the troubleshooting procedure on *page 167* under the heading "Your PC does not function properly after installing QEMM."

Also see the text file EX13FLOW.TEC in the \QEMM\TECHNOTE directory.

An Exception 6 or 13 inside of DESQview can often be fixed by using the Change a Program feature to allocate more memory to the program. If you are using DESQview/X, you can use DVP Manager to change the Maximum Program Memory Size. If that does not solve the problem, setting Protection Level to 3 will provide more information about what is causing the exception. Once you solve the problem, return Protection Level to 0.

The network does not load or work properly after you power on your system.

Your network adapter may be using an upper memory area that QEMM has filled with High RAM. If so, you will need to exclude the network adapter's upper memory addresses on the QEMM386.SYS line of CONFIG.SYS. There are two ways to find out what addresses to exclude:

- The network adapter's memory address range may display when the network driver initializes. Also, you can look in the network adapter's documentation to see if the addresses are listed or if there is a command you can type to display the addresses. Once you find out the adapter's upper memory addresses, edit CONFIG.SYS and use the EXCLUDE parameter to exclude those addresses on the QEMM386.SYS line (see the EXCLUDE parameter in **Chapter 7**).
- If you cannot find the network adapter's memory addresses as explained above, use the method described in step 4 on *page 169*.

Another possibility is that you have a bus-mastering network card (see the next troubleshooting problem).

A less likely possibility is that your network requires the QEMM parameter LOCKDMA:Y to prevent occasional crashes. See **Chapter 7** for more information on LOCKDMA:Y.

Your system locks up or gives you an "exception" error message during one of Optimize's reboots or when a device is loading high.

This problem may be related to a bus-mastering device (e.g., certain SCSI drive controllers or network adapters). Bus-mastering devices handle their own DMA (direct memory access) functions, bypassing the DMA controller on the PC's motherboard. These devices may not recognize QEMM's memory mapping.

The best solution is to contact the manufacturer of the device, who may have a VDS (Virtual DMA Services) device driver. VDS is an industry-wide specification supported by IBM, Microsoft, Quarterdeck, and many other hardware and software suppliers. A VDS driver requests memory addresses from the VDS manager (QEMM). If the VDS driver is for a hard disk controller, you can add the DISKBUF=1 parameter to the QEMM386.SYS line in the CONFIG.SYS to try to load the VDS driver high. Alternatively, you can load the VDS driver before QEMM.

If you cannot obtain a VDS device driver for a hard drive controller, try adding the parameter DISKBUF=2 to the QEMM386.SYS line in CONFIG.SYS. This parameter creates a buffer in unmapped memory for the disk drive to use when necessary. You can use a higher number to get better disk performance, but it will cost you conventional memory.

Your system reboots spontaneously when QEMM loads.

This problem is sometimes solved with the SHADOWRAM:NONE or the TOPMEMORY:N parameters. See **Chapter 7** for more information on these parameters. If neither of these parameters solve your problem, try the general troubleshooting procedure detailed above under the heading **Your PC does not function properly after installing QEMM**.

You have a problem with the Stealth ROM feature that goes away when you do not use Stealth ROM.

The text file STEALTH.TEC in the \QEMM\TECHNOTE directory outlines the steps to take in troubleshooting a problem with Stealth ROM.

Microsoft Windows does not run properly with QEMM installed.

The text file WINFLOW.TEC in the \QEMM\TECHNOTE directory gives a procedure for troubleshooting some common problems with Microsoft Windows and QEMM.

Your DOS program running in Microsoft Windows standard mode cannot find any EMS, VCPI, or DPMI memory.

In standard mode, Microsoft Windows takes all the memory normally controlled by QEMM for its own use. Non-Windows applications started in Microsoft Windows, which might otherwise use some of QEMM's memory using EMS or VCPI, will not find any, since Windows has taken all memory. If you want to run such programs under Windows, you can limit the memory Microsoft Windows 3.0 standard mode uses by using the EMBMEM parameter (see **Chapter 7**).

Windows 3.1 standard mode allocates memory through more than one interface, and so the EMBMEM parameter does not limit its memory allocation effectively. However, you can use the EMS.COM program that comes with QEMM to reserve memory before starting Windows in standard mode, and then free the memory from within a Windows DOS window. See **Chapter 11** for an example of how to use the EMS CREATE and EMS FREE commands for this purpose.

Your disk compression utility (e.g., Stacker, SuperStor) is preventing Optimize from completing properly.

See **Appendix B** for a list of technotes on the most popular disk compression utilities. The technotes are located in the \QEMM\TECHNOTE directory.

A graphics program has a corrupted or fuzzy display.

There is probably High RAM in an area that the graphics program needs to access for some of its display functions.

1. You may be able to correct the problem by excluding certain addresses from being used as High RAM. To exclude addresses, edit your CONFIG.SYS file and add the appropriate EXCLUDE (abbreviated "X") parameter to the QEMM386.SYS device driver line. See the next paragraph for suggestions on what addresses to exclude. Once you have added a parameter, save your CONFIG.SYS file, re-run the Optimize program (see **Chapter 3**), then try running your program.

If your program is running in a Super VGA or 1024x768 mode, try the parameter EXCLUDE=B000-B7FF. If you are running in standard VGA or any other mode, try EXCLUDE=FE00-FFFF or EXCLUDE=C000-C7FF. If these exclusions do not help, continue with step 2.

2. This could be a Stealth-related problem. Edit your CONFIG.SYS file and see if one of the Stealth ROM parameters (ST:F or ST:M) is on the QEMM386.SYS line. If neither of the ST parameters are there, skip to step 3.

   If you see the Stealth parameter, note whether it is ST:F or ST:M, then remove it. Reboot your PC and try your program again. If everything works, continue with the rest of this step; otherwise, skip to step 3.

   Run the Analysis procedure (see *page 125*) and when you get to step 2, use DEVICE=QEMM386.SYS ON MAPS=0 ST:x (substitute the appropriate letter, F or M, for x in ST:x). When running the Analysis, be sure to access the programs whose displays are corrupted. Exclude the addresses recommended by the Analysis.

3. If you are not using Stealth ROM, run the Analysis procedure (see *page 125*) and be sure to access the programs whose displays are corrupted. The Analysis report should give you information on what addresses need to be excluded. Edit the QEMM386.SYS line in CONFIG.SYS and exclude those areas.

You cannot use 1024x768 graphics modes after installing QEMM.

Some video cards use the upper memory area B000-B7FF (the monochrome text area) to display 1024x768 graphics, and QEMM usually makes this area available for High RAM or expanded memory mapping. If you are having problems running any graphics program in 1024x768 resolution, try adding the EXCLUDE=B000-B7FF parameter to the QEMM386.SYS device driver line in CONFIG.SYS.

After installing QEMM, your floppy drives do not work properly.

If you are using the ROM parameter on the QEMM386.SYS device driver line in CONFIG.SYS, try removing it. Some ROM code that manages floppy disks is sensitive to the speed at which it runs. If removing the ROM parameter helps the problem, you should be able to restore the ROM parameter if you use the EXCLUDE parameter to prevent QEMM from shadowing the part of the ROM that controls the floppy drives. See **Chapter 7** for more information on the ROM parameter.

If the problem persists, try removing any INCLUDE parameters that you have on the QEMM386.SYS line in CONFIG.SYS. You may be including a 4K region that is used when the floppy drives are accessed. Other parameters that may prevent floppy problems in some circumstances are the SHADOWRAM:NONE parameter and the MAPREBOOT:N parameter. See **Chapter 7** for more information on these parameters.

You have less conventional memory after installing QEMM than you did before.

Type **QEMM** and press **Enter** ↵ to see a Summary report. If you see the message **QEMM is not resident**, something has prevented QEMM from loading. Perhaps there is another memory manager in the CONFIG.SYS file, or perhaps there is an error in the QEMM386.SYS device driver line in CONFIG.SYS.

If the Summary report tells you QEMM is on, check the page frame address (also in the Summary report). By default, QEMM puts the page frame in upper memory (above 640K) to preserve as much free conventional memory as possible.

If the information displayed indicates Page Frame Address = 9000H or some other address that starts with a number from 0 to 9, QEMM has not found a contiguous 64K upper memory area to locate the page frame and is instead reserving 64K of conventional memory for programs that use expanded memory and require a page frame.

There are several possible ways to stop the page frame from using conventional memory. The most desirable is to use QEMM's Stealth ROM parameter, which creates more usable memory above 64K. Run OPTIMIZE /STEALTH to optimize and enable Stealth ROM.

Another solution is to consult your hardware manuals and reconfigure your system so that your adapters are at locations that do not prevent QEMM from putting the page frame in upper memory.

Finally, the last resort solution is to eliminate the page frame altogether by adding the parameter FRAME=NONE to your QEMM386.SYS device line. This saves the memory, but leaves no page frame for programs that use EMS, so these programs will not be able to use EMS.

The Suspend/Resume feature on your laptop or notebook computer does not work when QEMM is loaded.

See the information on the SUSPENDRESUME parameter in **Chapter 7**.

Your system hangs when you try to warm boot with QEMM installed.

QEMM's Quickboot feature may be incompatible with a piece of hardware or software on your system. The BOOTVIA19:Y may help if a TSR or device driver on your system needs to be notified when a warn boot occurs. If this does not work, you can disable the Quickboot feature with the BOOTENABLE:N parameter.

If you have an Intel Inboard-AT system and you cannot warm boot your system, use QEMM's TRAP8042:Y parameter. See **Chapter 7** for more information on the QEMM parameters.

Your system hangs when Manifest starts.

A few systems hang when programs attempt to read from I/O port 200, which is usually used by game ports. Manifest reads I/O port 200 to detect the presence of game ports. You can give Manifest the /G switch to prevent it from trying to detect a game port on your system. If the /G switch causes Manifest to run properly on your system, you should report this problem to the manufacturer of your computer.

High-speed communications fail when QEMM is active.

If high-speed communications stop suddenly, try QEMM's TASKS=xx parameter with xx set to a number higher than QEMM's default of 16. See **Chapter 7** for a discussion of the TASKS parameter.

QEMM posts an "Unknown MCA Adapter ID" message when loading on your Micro Channel system.

QEMM's MCA.ADL file contains no information about one of the adapters on your system. If your adapter uses no RAM or ROM in upper memory, or if QEMM can detect this RAM or RAM without the help of the MCA.ADL file, then you will experience no problems because of the missing MCA.ADL entry. In this case, you can tell QEMM386.SYS not to pause for the error message by using its PAUSEONERROR:N parameter, and simply ignore the message.

If you are experiencing problems in addition to the error message, contact Quarterdeck Technical Support to have information on your adapter added to your MCA.ADL file.

If you experience problems loading a command processor (e.g., 4DOS or NDOS) into upper memory.

First be sure that you have followed the instructions in the command processor's manual for loading it into upper memory. Also be sure you are using QEMM's DOS-Up feature to load the command processor high. To do that, run QEMM Setup, select **Enable or disable DOS-Up**, select **Partial**, then be sure the **COMMAND.COM** selection is set to **Yes**. Save your configuration and run Optimize.

If you still experience problems loading the command processor into upper memory, you will have to load it into conventional memory by following the instructions in the command processor's manual for loading it into conventional memory. Run QEMM Setup, select **Enable or disable DOS-Up**, select **Partial**, then be sure the **COMMAND.COM** selection is set to **No**. Save your configuration and run Optimize.

You are having problems accessing memory above 16 megabytes.

See the section on SCANMEM.COM in **Chapter 12**.

Finding additional troubleshooting information.

We have provided additional troubleshooting information in several "README" files. See **Appendix B** for a list of these files and what information they contain.

When starting up your computer you see the following message:

QEMM386: Disabling Stealth because QEMM could not locate the ROM handler for INT XX

There are three possible reasons for this message:

1. You are loading a driver before QEMM which is grabbing interrupt XX.

   If this is the case, try loading the driver in question after QEMM. If it must be loaded before QEMM, load HOOKROM.SYS before you load this driver (see **Chapter 12** for information

on HOOKROM.SYS). Assuming that QEMM is installed in a directory called QEMM on your "C" drive, the new line would look like this:

```
DEVICE=C:\QEMM\HOOKROM.SYS
```

2. A ROM is loading a handler for interrupt XX into RAM.

   Add the parameter XSTI=xx (where xx is the number of the interrupt reported in the message) to the QEMM386.SYS device line in CONFIG.SYS, then add the appropriate EXCLUDE parameter to this same line to keep QEMM from mapping over the portion of the address space where the ROM handler for interrupt XX resides.

   You will need to add an address range to the EXCLUDE parameter; for information on the EXCLUDE parameter, see **Chapter 7**. To find the appropriate address range to exclude, first, locate all your ROMs. You can do this by looking at Manifest's First Meg/Overview screen. If you have a non-Micro Channel machine and VGA video, you typically have a system ROM at F000-FFFF and a video ROM at C000-C7FF. If you have a PS/2 or other Micro Channel machine, you typically have one ROM at E000-FFFF. Add-on devices, such as some disk controller cards and network cards, may also have ROMs, which you must exclude as well.

   Next, edit the QEMM386.SYS line in your CONFIG.SYS file and add excludes for all your ROMs. Also add the FSTC parameter to this line.

   A typical QEMM line for a non-Micro Channel machine is:

```
DEVICE=C:\QEMM\QEMM386.SYS RAM
ST:M XSTI=XX X=F000-FFFF X=C000-C7FF
FSTC
```

   On a PS/2 or most Micro Channel machines, the line will look like this:

```
DEVICE=C:\QEMM\QEMM386.SYS RAM
ST:M XSTI=XX X=E000-FFFF FSTC
```

   In the above examples, XX is replaced with the interrupt reported in the QEMM error message.

   Reboot your computer. Stealth ROM should work this time. Use your computer for a while, then look at the QEMM/Analysis screen of Manifest.

   You will see a chart that looks something like this:

```
   n=0123 4567 89AB CDEF
0n00 OOOO OOOO OOOO OOOO
1n00 OOOO OOOO OOOO OOOO
2n00 OOOO OOOO OOOO OOOO
3n00 OOOO OOOO OOOO OOOO
4n00 OOOO OOOO OOOO OOOO
5n00 OOOO OOOO OOOO OOOO
6n00 OOOO OOOO OOOO OOOO
7n00 OOOO OOOO OOOO OOOO
8n00 OOOO OOOO OOOO OOOO
9n00 OOOO OOOO OOOO OOOO
An00 OOOO OOOO OOOO OOOO
Bn00 OOOO OOOO OOOO OOOO
Cn00 IIII IIII OOOO OOOO
Dn00 OOOO OOOO OOOO OOOO
En00 OOOO OOOO OOOO OOOO
Fn00 IIII IIII OOII IIIO
```

   An "O" indicates a portion of the address space that has been accessed and an "I" indicates a portion of the address space that has not been accessed. You must exclude that portion of the address space in the excluded ROMs where you now see Os.

In this example (which presumes that the ROMs were located from C000-C7FF and F000-FFFF), the appropriate exclude is X=F800-F9FF, an 8K portion of the address space. This is the portion of the address space where the ROM handler for the interrupt XX resides. Our QEMM line, with appropriate excludes, would read as follows:

```
DEVICE=C:\QEMM\QEMM386.SYS RAM
ST:M XSTI=XX X=F800-F9FF
```

The FSTC parameter is used only during this analysis process and should be removed afterward. Because the last 64 bytes of the First Meg address space (in FFFC-FFFF) is still addressed directly with Stealth ROM, the last 4K piece of the QEMM/Analysis screen will always have an "O" in it, whether an exclude is appropriate or not.

Also, this procedure is *not* used to find INCLUDES in portions of the address space that are not occupied by Stealthed ROMs. If you want to experiment with INCLUDES (to gain additional High RAM) you must perform a complete analysis as described in the section "The Analysis Report" in **Chapter 9**).

It is possible that there are no "O"s at all: this is because the ROM handler for interrupt xx has been replaced by a new interrupt handler and the one in the ROM is not being accessed at all. No exclude is necessary in this case.

In some case, it may be possible to reconfigure your system in such a way that the ROM no longer redirects an interrupt into RAM. This is the case with the Invisible Network. The procedure above will be unnecessary in this case.

3. You are using a computer which was upgraded to an 80386 with an add-in board, such as the Intel Inboard-AT PC. In this case, add the following parameters to the QEMM device line in your CONFIG.SYS file: XSTI=70 XSTI=74 XSTI=75 XSTI=76

   A typical QEMM line would look like this:

```
DEVICE=C:\QEMM\QEMM386.SYS RAM
ST:M XSTI=70 XSTI=74 XSTI=75 XSTI=76
```

The hard disk becomes unreadable immediately after DOS-UP.SYS loads during the Optimize process or during bootup, and the message, "Device not found," displays when the system attempts to load whatever driver follows DOS-UP.SYS.

Load the QEMM program INT13FIX.SYS in the CONFIG.SYS file, immediately before the QEMM386.SYS line (and after DOSDATA.SYS and any other programs loaded before QEMM386.SYS). Some UltraStor disk controllers are known to need INT13FIX.SYS. See **Chapter 12** for information on INT13FIX.SYS.

Optimize or the bootup process fails with a "Configuration too large for memory" message, or simply crashes, on certain system with an Adaptec 1542c disk controller.

Place the following line in the CONFIG.SYS file, immediately before the QEMM386.SYS line (and after DOSDATA.SYS and any other programs loaded before QEMM386.SYS):

```
DEVICE=C:\QEMM\INT13FIX.SYS/STACKSIZE=3
84
```

Contacting Quarterdeck Technical Support

If you need further assistance, you can contact Quarterdeck Technical Support. See the Passport booklet included with QEMM for information on technical support.

If you are calling from the United States and you have a touchtone phone, we suggest you try 1-800-ROBOTECH, Quarterdeck's toll-free, automated technical support hotline. 1-800-ROBOTECH can assist with the most common technical questions and offer a variety of solutions. Navigation through 1-800-ROBOTECH is accomplished by pressing numbers on your phone's keypad to jump directly to the topic that you are interested in hearing about. The system will instruct you every step of the way. Call 1-800-ROBOTECH (1-800-762-6832), toll free, 24 hours a day, 7 days a week, including Holidays.

If you contact us by mail, fax or on one of our BBS systems, please include the following information:

- Your Quarterdeck customer VIP number which you receive when you register your copy of QEMM with Quarterdeck.
- The version number and serial number of QEMM. To find these out, type **QEMM /REG** at the DOS prompt.
- If you are contacting us by mail or fax include a printout from Quarterdeck's Manifest. If you are using the DOS version of Manifest, press F2 to print, and select **All Manifest** from the **What to Print** portion of Manifest's print menu. If you are using the Windows version, select **Print** from the File menu, then select **All Manifest**. If you have other important hardware in the system, or if Manifest's list is incomplete, please include any additional information you think may help us diagnose your problem.
- If you cannot run Manifest, print out your CONFIG.SYS and AUTOEXEC.BAT files, and write down what hardware (include the make and model) and software (include the version) you are using.
- Give a precise description of the problem that is occurring, and the exact text of any error messages. Describe in detail the results of your troubleshooting efforts.
- Please tell us how to respond to you via mail, fax or one of the other methods we support. See your Quarterdeck Passport booklet for information on contacting Quarterdeck Technical Support.

If you are contacting a technical support representative by telephone:

- Be at your computer.
- Please gather the information listed above.
- When you contact our technical support representative, you need only give your customer VIP number or product serial number and a brief description of your hardware, software and the problem you are encountering. If the support technician requires additional information, he or she will ask for specific details.

NOTES:

B QEMM's Technical Bulletins

QEMM includes several technical bulletins that provide detailed information on a variety of subjects. Some of these "technotes" contain troubleshooting information, others contain information on fine-tuning your system and others contain technical background information on QEMM features and how they relate to other products (e.g., DOS 6). The technotes are ASCII files located in the \QEMM\TECHNOTE directory on your hard disk. You can read the technotes using the QEMM Setup program or by loading them into a text editor or word processor (if you use a word processor, be sure to load the file as an ASCII file).

To read technotes using the DOS version of QEMM Setup:

- **Type QSETUP and press Enter ↵.**
- **When QEMM Setup's opening screen displays, press Enter ↵.**
- **At the main menu, select View QEMM hints, technotes and READ.ME.**
- **When the list of topics displays, select View technical notes.**
- **You will see a list of topics, and you can select the topic you want. The list is several screens long. Use PageDown and PageUp to scroll through the list.**

Below is a summary of the topics discussed in the technotes and the names of the files on disk.

To read technotes using the Windows version of QEMM Setup:

- **Go to the QEMM group (or QEMM folder if you are using SideBar).**
- **Select Notes.**

Operating Systems

QEMM is fully compatible with DOS versions 3.x and higher, DR DOS versions 5.0 and higher and Novell DOS 7. For information on optimally configuring your system with your operating system, see the appropriate technote: MSDOS6.TEC, DOS5.TEC, DRDOS6.TEC, NOVELL7.TEC.

Microsoft Windows 3.0/3.1

QEMM is fully compatible with Microsoft Windows 3.0 and 3.1. If you experience any problems running Windows or a Windows application, we suggest you follow the troubleshooting suggestions in WINFLOW.TEC. In most cases, this guide will help you pinpoint the problem and resolve it.

QEMM's Stealth ROM Feature

Chapter 5 of this manual describes QEMM's Stealth ROM feature. For a more detailed explanation of Stealth ROM, see STLTECH.TEC.

If you experience a problem while running a program or accessing a hardware device and you believe the problem may be related to Stealth ROM, see STEALTH.TEC for troubleshooting suggestions.

Maximizing your DESQview and DESQview/X Windows

To help you get the most out of your PC's memory and run large programs from within DESQview and DESQview/X, we have written two technical bulletins. WINSIZE.TEC provides tips on increasing the size of DESQview windows, and MAXWINDO.TEC does the same for DESQview/X.

Bus-Mastering Adapters

Certain hard disk controllers and other adapters use a technique called bus mastering to speed up access times. While this technique does speed up your hard drive, it can cause conflicts with 386 memory managers (such as QEMM), particularly when the memory manager attempts to load device drivers or memory resident programs into upper memory. Fortunately, there are solutions to the problems created by bus-mastering hardware. For a detailed explanation of bus mastering and what can be done to resolve the problems associated with its use, see BUS-MAST.TEC.

Disk Compression Utilities

If you are using SuperStor version 2.04 or SuperStor Pro, refer to technical bulletin SSTOR.TEC for information on using QEMM's Optimize program with SuperStor.

If you have Stacker 2.01 or 3, you will want to read STACKER3.TEC for tips on using your program with QEMM. If you have version 2.0, you should contact STAC Electronics for an upgrade.

STACKER4.TEC contains information about using QEMM with Stacker version 4.

If you are using XtraDrive, you should read XTRADRV.TEC for information on using this program with Quarterdeck products.

Parity Errors

Usually, a parity error indicates a hardware problem of some kind. PARITY.TEC explains why these errors occur and tells you how to determine if you have defective memory chips or other hardware.

Troubleshooting QEMM

PRODUCTS.TEC describes solutions to compatibility issues that may arise when using QEMM with specific hardware and software. Before beginning any troubleshooting procedures, we suggest you check PRODUCTS.TEC for references to your specific hardware and/or software configuration.

Other problem solving technotes are:

- QEMMFLOW.TEC - A thorough, advanced QEMM troubleshooting guide.
- EXCLUDE.TEC - Finding specific areas of memory that need exclusions.
- EXCEPT13.TEC - Detailed explanation of Exception 6 and 13 errors.
- EX13FLOW.TEC - Troubleshooting the cause of an Exception 6 and 13 error.

Technical Support

CONTACT.TEC tells you how to contact Quarterdeck for technical support via phone, fax, letter, or various electronic services. If you are unable to find the answer to your question or problem in this manual, in the technotes included on your disk, or in QEMM Setup's Help screens, refer to CONTACT.TEC for instructions on contacting our Technical Support Department.

Additional QEMM Technotes

Quarterdeck has additional technotes available on request. You can obtain these technotes from various sources: via the Quarterdeck BBS (310-314-3227), by calling QFAX (our automated FAX-back service, at 310-314-3214), on CompuServe (go QUARTERDECK, library 2), or BIX. Many technotes are also available on large local BBSs, often in "DESQview" or "multitasking" file areas. Note that under CompuServe and BIX, some of the technotes may be zipped; so, if you cannot locate the file you want, try using the extension .ZIP instead of .TEC.

Subject	Filename	Document # when using QFAX
Unknown MicroChannel adapter	MCA.TEC	223
LAN Manager problems	LANMAN.TEC	258
PROTMAN.SYS or PROTMAN .DOC	PROTMAN.TEC	183
Shift, Ctrl, or NumLock keys appear to be engaged when not pressed	IA.TEC	221
Pathworks/DECNET	DECNET.TEC	220
IBM's PC support for 5250	AS400.TEC	212
Double Disk	DBLDSK.TEC	218
LAN Manager	LANMAN.TEC	258
Master list of Quarterdeck technotes	MASTER.TEC	100
Third-party books about Quarterdeck products	BOOKS.TEC	132
QEMM and CD ROM drives	CD-ROM.TEC	261

C Memory Specifications Supported by QEMM

QEMM supports a variety of industry-standard specifications that are designed to remove incompatibilities between programs, or between a program and hardware. Below is a brief description of each of these specifications.

- **EMS (Expanded Memory Specification)** lays down the guidelines for the use of expanded memory. Expanded memory is memory outside of the first megabyte of memory addresses that can be made to appear in the first megabyte. Its purpose is to give programs access to more memory than exists in the one-megabyte address space to which real-mode programs are limited. Most programs use EMS memory to store large amounts of data; multitasking environments like DESQview and DESQview/X use it to run multiple large programs at the same time. The Expanded Memory Specification describes programming calls that programs can make and that an expanded memory manager (EMM) like QEMM must service. For copies of the EMS documentation, call (800) 548-4725 or write Intel Literature JP26, 3065 Bowers Ave., P.O. Box 58065, Santa Clara, CA 95051-8065.

- **XMS (Extended Memory Specification)** governs access to three different areas of memory: extended memory, which is memory addressed above the one-megabyte boundary and accessible only in protected mode; upper memory, which is the "reserved area" between 640K and one megabyte; and the High Memory Area, the first 64K of extended memory, and the only region of extended memory that can be accessed in real mode.

These three areas correspond to the three kinds of memory that XMS programs obtain from an XMS manager: Extended Memory Blocks (EMBs), Upper Memory Blocks (UMBs), and the High Memory Area (HMA). EMBs are used by real-mode programs to store large amounts of data, or by protected-mode programs for both code and data. UMBs are used by real-mode programs to place relatively small amounts of code and data above 640K, thereby reducing their footprint below 640K. The HMA, which can only be used by specially written programs, is generally allocated by operating systems or environments to reduce their footprint

below 640K. It is common for one XMS manager to provide EMBs and the HMA and another to provide UMBs; QEMM provides all three XMS services. For copies of the XMS documentation, call (800) 227-4679 or write Microsoft Corporation, One Microsoft Way, Redmond, WA 98052.

- **VCPI (Virtual Control Program Interface)** makes it possible for a program to go into protected mode when a 386 memory manager like QEMM has already taken control of the system. Without such a specification, programs would cause system errors when they tried to go into protected mode with a 386 memory manager active. The protected mode application, known as the VCPI client, makes programming calls to QEMM, the VCPI host, to allocate memory and to switch the processor's mode. A VCPI client takes control of the system from the VCPI host when it is operating in protected mode, and hands control back to the VCPI host when it switches out of protected mode. For copies of the VCPI documentation, call (617) 661-1510 or write Phar Lap Software, Inc., 60 Aberdeen Ave., Cambridge, MA 02138.

- **DPMI (DOS Protected Mode Interface)** is, like VCPI, a specification to allow the coexistence of protected mode programs and 386 memory managers. Unlike VCPI, DPMI dictates that the DPMI host be in control of the system at all times. DPMI thus gives more power to the host, whereas VCPI gives more power to the client. QEMM's support for DPMI is in the QDPMI program, which is loaded automatically when you install QEMM. For copies of the DPMI documentation, call (800) 548-4725 or write Intel Literature JP26, 3065 Bowers Ave., P.O. Box 58065, Santa Clara, CA 95051-8065.

- **VDS (Virtual DMA Services)** makes it possible for bus-mastering devices to coexist with 386 memory managers like QEMM. Bus-mastering devices (like some hard disk controllers and network adapters) do not use the system's DMA (Direct Memory Access) hardware, which provides a way to transfer data back and forth from devices to memory while the main processor is busy with other work. Instead, the bus-mastering device uses its own hardware to perform this same function. By bypassing the well-documented DMA interface, the bus-mastering device circumvents QEMM's memory mapping, and is likely to send data to the wrong locations in memory. VDS provides a programming interface that lets devices ask the VDS host (QEMM) for the correct address in memory to which to transfer data. VDS can also be used

to avoid incompatibilities between QEMM and devices that use the conventional DMA hardware in unusual and unsupported ways. For copies of the VDS documentation, call (800) 227-4679 or write Microsoft Corporation, One Microsoft Way, Redmond, WA 98052.

NOTES:

D Glossary

A

A20 - A hardware address line that can be enabled or disabled under program control. The A20 line is important in the management of extended memory and the HMA. Programs usually enable the A20 line when using extended memory and the HMA, and disable it otherwise.

Adapter RAM - Memory regions that have RAM mapped into them by adapter cards.

Address Space - The range of memory addresses that the processor can reference. PCs running in real mode have an address space of one megabyte. An 80386 (or higher) processor running in protected mode has an address space of up to four gigabytes.

Alternate Maps - See Real Alternate Maps.

AUTOEXEC.BAT - The batch file automatically executed by DOS when the system boots up. This file typically specifies the PATH and PROMPT, and loads whatever TSRs the user has specified.

B

BIOS - Basic Input Output System. A program, contained in the system ROM, that provides low-level, basic system services.

BIOS Data Area - The data area, located 1K from the bottom of conventional memory, that is used by the BIOS to keep track of the state of the keyboard, video adapter, and other standard hardware.

Buffers - Storage areas used by DOS to hold data on its way from a disk to memory or back. Each buffer uses about 528 bytes of memory. Increasing the number of buffers tends to speed up file operations at the cost of decreasing available memory. Though "buffer" is a general term for any storage area that holds data on its way from one place to another, the term is commonly used to refer to DOS's disk buffers.

BUFFERS - A CONFIG.SYS statement that defines the number of buffers placed by DOS in memory.

Bus Mastering - A technique by which some devices use their own hardware, instead of the processor or the PC's DMA chip, to transfer information back and forth to memory. Bus mastering will cause incompatibilities with 386 memory managers unless the bus-mastering device and the memory manager both support VDS, or unless some other measures are taken to insure compatibility.

C

CMOS - A special memory area that is used to store configuration information needed at system boot time. A battery maintains the contents of this memory while power is off.

Command Line Switch - Words or symbols on a command line that follow the program name and specify certain commands or options to the program.

COMMAND.COM - The standard command processor used by DOS. This is the program responsible for displaying the DOS prompt and processing DOS commands (COPY, DIR, etc.).

CONFIG.SYS - The file that defines the initial configuration of DOS. It typically specifies the number of DOS resources to be allocated and device drivers to be loaded.

Conventional Memory - Memory starting at 0K and used by DOS to run real-mode programs. Conventional memory is usually 640K in size.

Coprocessor - An optional auxiliary processor used to speed up complex math operations.

D

DESQview - An operating environment created by Quarterdeck Office Systems that allows off-the-shelf DOS programs to multitask in separate windows on the screen. It also provides a menu-driven interface to DOS, a macro facility, and the ability to transfer information from one DOS program to another.

DESQview/X - A graphical operating environment created by Quarterdeck Office Systems. In addition to the features of the text-based operating environment DESQview, it provides full support for the X Window System and can turn DOS text programs and Microsoft Windows programs into X clients.

DEVICE - A CONFIG.SYS statement that tells DOS to load an add-on device driver at system startup time.

DevDriven - A term used in Manifest to refer to certain drives, including those created by disk compression utilities such as SuperStor, Stacker, and DoubleSpace, which do not exist unless a device driver is loaded in the CONFIG.SYS file during bootup.

Device Driver - A piece of software used to communicate with an installed hardware device (such as a printer, video adapter, disk drive, or memory card). It is either built into DOS or loaded using the DEVICE statement in CONFIG.SYS.

DMA - Direct Memory Access. A feature of the PC's architecture that allows the transfer of information from devices to memory and back without using up processor time. Some devices use DMA and some transfer information through the processor.

DOS Data - The data area used by DOS. It normally occupies low conventional memory, but can be placed in the upper memory area by the DOS-Up feature.

DOS Extender - A program that is used by protected-mode applications to handle the task of switching back to real mode when necessary to retain compatibility with DOS and the BIOS. Applications are often built with commercial DOS extenders made by other companies.

DOS Kernel - The two program files that contain the DOS code. These files are called IBMBIO.COM and IBMDOS.COM in IBM DOS, and are called IO.SYS and MSDOS.SYS in MS-DOS.

DOS Protected Mode Services - See DPMI in **Appendix C**.

DOS Resources - Resources that DOS allocates in quantities that can be controlled by CONFIG.SYS statements: BUFFERS, FCBS, FILES, LASTDRIVE, and STACKS. In the absence of such statements, DOS assigns default values for each resource.

DOS-Up - A QEMM feature that loads pieces of DOS (including DOS data, DOS resources, and COMMAND.COM) into upper memory, saving 7K-70K of conventional memory.

DPMI - DOS Protected Mode Services. See **Appendix C**.

Environment - 1) A block of memory that contains the current PATH, PROMPT, COMSPEC, and any other variables defined using the SET command. Programs often look in the environment for directives, set by the user with the SET command, on how to configure themselves. 2) A program, such as Quarterdeck's DESQview or Microsoft Windows, that resides on top of DOS and whose purpose is to run other programs in ways that enhance the usability and functionality of the operating system.

Exclude (X) - On the Manifest QEMM Analysis screen, Exclude (X) refers to areas of memory that have been accessed by a program, but that QEMM did not expect to be accessed. You should use QEMM's EXCLUDE parameter to prevent QEMM from using these areas for High RAM or expanded memory mapping.

Expanded Memory - Memory that can be mapped into the first megabyte of address space in 16K chunks called pages. Programs manipulate expanded memory by making calls to an expanded memory manager (EMM).

Expanded Memory Specification: See EMS in **Appendix C**.

EMB - Extended Memory Block. See XMS in **Appendix C**.

EMS - Expanded Memory Specification. See **Appendix C**.

EMS 4.0 - The current standard expanded memory specification which unifies the two previous standards, EMS 3.2 and EEMS, and added support for multiple real alternate maps, mapping conventional memory, and mapping more than 64K of upper memory. It also added various software features that make it easier to write expanded memory programs.

Exceptions - Conditions caused by specific processor instructions that cause 80386 (and higher) processors to transfer control to routines known as exception handlers. When QEMM reports an exception, it generally means that an error has occurred that jeopardizes the system's integrity.

Extended BIOS Data Area - A data area, usually 1K in size, located at the top of conventional memory on IBM PS/2s and many other machines. By default, QEMM moves this data area into High RAM or to low conventional memory.

Extended Memory - Memory addressed above 1024K (one megabyte).

Extended Memory Block - A block of extended memory accessed through XMS calls. Extended memory blocks are used by real-mode programs to store data, or protected-mode programs to store code and data.

Extended Memory Specification - See XMS in **Appendix C**.

F

FCBS - A CONFIG.SYS statement that specifies the number of file control blocks kept track of by DOS.

File Control Blocks - The data structures used in DOS 1 to access files, rarely used in modern programs.

File Handles - The data structures used by modern DOS programs to access files, introduced in DOS 2.

FILES - A CONFIG.SYS statement that specifies the maximum number of file handles that can be open simultaneously.

H

Handle - A shortcut value used to refer to a particular item. For example, DOS files are assigned a handle when opened so that a program need not refer to the full name each time. Expanded memory handles are used to refer to groups of expanded memory pages. XMS handles are used to refer to blocks of extended memory.

Hexadecimal Numbering - A base 16 number system in which a digit has a value 16 times greater than the digit to its right. In hexadecimal numbering, the letters A, B, C, D, E, and F are enlisted to represent the values above 0 through 9. Hexadecimal numbering is well suited to clear and economical representation of values stored in computers.

High Memory Area - The area from 1024K to 1088K (the first 64K of extended memory), accessed through XMS calls. It is important because it is the only extended memory that can be accessed in real mode; some operating systems or environments load some of their code into the HMA to reduce their overhead in conventional memory.

High RAM - QEMM's term for RAM mapped into portions of the upper memory area. Device drivers and TSRs may be loaded here to minimize conventional memory overhead.

HMA - See High Memory Area.

I

Include (I): On the Manifest QEMM Analysis screen, Include (I) refers to areas of upper memory that have not been used by a program, but are not currently available for High RAM or expanded memory mapping. You can use QEMM's INCLUDE parameter to tell QEMM to use these areas.

Interrupt Vector Table - A table of pointers to the locations of interrupt handlers, routines that service hardware events and software requests. The first 1024 bytes (1K) of conventional memory are devoted to storing interrupt vectors.

Interrupts - Forced transfer of control to special service routines known as interrupt handlers. Software interrupts occur when programs issue certain instructions that request services from the system. Hardware interrupts are triggered at random times in response to signals from hardware, and are handled by routines that respond appropriately to the hardware event. After an interrupt, control returns to the executing program.

I/O Port Address - An address that can be used to access a piece of hardware with the processor's IN and OUT instructions. I/O addresses are part of an address system that is separate from the one used for memory.

K

K - The abbreviation for a kilobyte (1024 bytes).

Keyboard Buffer - The place in the BIOS data area where the BIOS stores keystrokes as they are typed.

L

LASTDRIVE - A CONFIG.SYS statement that specifies the drive letter of the last drive that DOS should support.

LOADHI - Two QEMM programs used to load other programs into High RAM. The LOADHI.COM program loads TSRs into High

RAM, and the LOADHI.SYS program loads device drivers into High RAM.

M

Mappable Areas - Portions of the address space into which the expanded memory manager can place expanded memory upon request.

Mappable Pages - 16K segments of address space in the first megabyte, aligned on 16K boundaries, into which an expanded memory manager can map expanded memory. Manifest's Expanded Pages screen identifies mappable pages with a + sign. You can also create High RAM in mappable areas by using the RAM parameter on QEMM. Areas in which you have created High RAM will show up as Unmappable (U) on this screen.

Mapping - The process of making a portion of memory (on 386 PCs, usually extended memory) appear at a different address.

MB - The abbreviation for megabyte (1,024 kilobytes or 1,048,576 bytes).

Memory Address - A number that can be used to access a piece of memory. Memory addresses are broadcast throughout the system on the bus, and the hardware that controls the relevant piece of memory responds to the broadcast. There may or may not be a piece of memory corresponding to any particular memory address. However, QEMM can use the power of the 80386 (and higher) processor to make memory appear at any location in the address space.

Micro Channel Architecture (MCA) - A bus design implemented by IBM in its PS/2 line of computers. One of the advantages of Micro Channel architecture is that it allows easy identification of what adapters are installed.

Mode - See QEMM Mode.

N

Network Adapter - A hardware card that provides the interface between your processor and a network. This hardware often reserves portions of the upper memory area for its own use.

O

OFF - A QEMM mode in which the processor remains in real mode. Many QEMM features are not available when QEMM is OFF.

OK - On the Manifest QEMM Analysis screen, OK (O) refers to areas of memory that QEMM thinks are properly identified, and that should not be the subject of EXCLUDE or INCLUDE parameters.

ON - A QEMM mode in which the processor is in either virtual-8086 mode or protected mode. QEMM must be ON to provide most of its features.

Optimize - A QEMM program that determines how best to load TSRs, drivers, and parts of DOS and QEMM into High RAM. Optimize is run automatically during the installation process, and should be run manually if you change your system configuration after QEMM installation.

P

Page Frame - An EMS term describing the mappable area, 64K in size, that is used by most EMS programs for mapping expanded memory. The page frame is usually located in the upper memory area.

Parallel Ports - Hardware ports used for communication with parallel devices such as printers, accessed through I/O port addresses that can be found in the BIOS data area.

Parity Errors - Errors signalled by a hardware subsystem of the PC that continuously checks the integrity of the contents of memory. A parity error usually indicates faulty hardware.

Processor - The chip that is the central control unit of your PC and determines many of its capabilities. QEMM runs on 80386 and higher processors.

Protected Mode - A collective name given to several advanced modes of 80286 and higher processors. In general, the term refers to processor modes in which all of extended memory can be accessed. DOS and the BIOS cannot run in protected mode; however, programs that use a DOS extender can run in protected mode while still maintaining compatibility with DOS and the BIOS. See also Real Mode and Virtual-8086 Mode.

Q

QDPMI - QEMM's DPMI driver, placed in the CONFIG.SYS file by default by QEMM's Install program.

QEMM Mode - A category on the QEMM Overview screen of Manifest that refers to the current state of QEMM386.SYS. When there is High RAM, ROM mapping, memory backfilling, or video filling, QEMM's mode is ON and may not be turned OFF. Otherwise, QEMM's mode will usually be AUTO/ON or AUTO/OFF, depending upon whether expanded memory is in use. If QEMM's mode is OFF (without the AUTO), then expanded memory is not accessible until you change the mode using the QEMM.COM program.

QEMM Status - A category on the QEMM Overview screen of Manifest that shows you the current status of QEMM386.SYS. Manifest may indicate that QEMM is configured with High RAM or Mapped ROM, that memory is backfilled or video filled, or that expanded memory is being used.

QSETUP - A QEMM program that provides an easy way to turn QEMM's features on and off, as well as access to troubleshooting tips and an editor for CONFIG.SYS and AUTOEXEC.BAT.

R

RAM - Random Access Memory. This is the memory in your machine into which programs and data can be loaded. Its contents are lost when power is turned off. It is different from ROM in that the contents of RAM can be changed.

Rammable Areas - Vacant areas in upper memory that are large enough for QEMM to fill with High RAM, but too small to fill with a page of expanded memory (which must occupy a 16K area, aligned on a 16K boundary). You must add the RAM switch to the QEMM386.SYS line to fill a rammable area with High RAM.

Real Alternate Maps - A full description of the EMS mapping state, stored in hardware registers. When real alternate maps are available, the state of EMS mapping can be changed quickly, improving performance in a multitasking environment.

Real Mode - The processor mode common to all Intel microprocessors and the mode for which DOS and the BIOS are

written. In real mode, only the first megabyte (1024K) of memory is addressable. See also Protected Mode and Virtual-8086 Mode.

Region - In QEMM parlance, a contiguous area of High RAM.

Reserved - As used on several Manifest screens, "reserved" refers to a memory area or number that has no standard function, but has been reserved for a future use. Some "reserved" functions are currently used in different, unstandardized ways by different manufacturers.

ROM - Read-Only Memory. Memory that is fixed in content and cannot be changed. The contents of ROM memory are not lost when the power is turned off. ROMs generally occupy addresses in upper memory. The BIOS and video services are among the programs contained in ROM.

ROM Holes - In QEMM parlance, areas of the System ROM that are not in use and that QEMM has turned into mappable, rammable, or High RAM areas. QEMM only reclaims unused ROM areas if the ROM is not being Stealthed.

S

Serial Ports - Hardware ports used for communication with serial devices such as modems, accessed through I/O port addresses that can be found in the BIOS data area.

Shadow Memory - 384K or RAM that is used on some PCs to speed up the operation of ROMs. It is typically inaccessible for any other purpose, but QEMM can often reclaim unused shadow memory and add it to QEMM's memory pool.

ShadowRAM - Chips and Technologies' implementation of shadow memory.

Split ROM - A piece of ROM that occupies only a portion of a 4K area aligned on a 4K boundary. Such areas cannot be Stealthed, or sped up with the ROM parameter.

Squeeze - A feature of Optimize that lets TSRs and drivers temporarily use memory areas that are excluded, that are part of the EMS page frame, or that belong to ROMs or Adapter RAM. Squeeze lets some programs load into High RAM that would ordinarily not fit in any High RAM region.

Stack - An area in a program's memory used to store values on a first-in-last-out basis. PC processors provide instructions and registers that, used properly, automatically manage stacks for programs.

STACKS - A CONFIG.SYS statement that specifies the number and size of the stacks DOS uses for servicing hardware interrupts.

Stealth D*Space - A feature of QEMM that eliminates almost all of the conventional memory and upper memory overhead of DOS's DoubleSpace and DriveSpace disk compression utilities. Stealth D*Space places the disk compression drivers into the EMS page frame when they are needed.

Stealth ROM - A feature of QEMM that frees up the address space that ROMs would otherwise use, making it available for High RAM or expanded memory mapping. Stealth ROM places ROMs into the EMS page frame when they are needed.

Suspend/Resume - The feature of some laptop and notebook computers that allows the system to shut down (saving power and battery life) and resume operation with the state of the system preserved. This feature is known by various names, depending on the system manufacturer.

System ROM - The ROM, usually located at address F000, that contains the BIOS, which is responsible for initializing and servicing requests to standard hardware (disk drives, keyboard, simple video adapters) attached to the PC.

T

Top Memory - 384K of memory located just below the 16 megabyte point on Compaqs and some other systems, reserved for speeding up ROMs and for other system functions.

TSR - Terminate-and-Stay-Resident program (also called memory resident program). A program that loads and then returns to the DOS prompt. TSRs monitor some system interface so that they can perform a function while other programs are being used. TSRs can sometimes be "popped up" over a currently running program when the user hits a particular key or key combination. QEMM's LOADHI.COM program can load TSRs into High RAM.

U

UMB - See Upper Memory Block.

Unassigned Memory - As reported on Manifest's QEMM Memory screen, unassigned memory is extended memory that QEMM makes unavailable to other programs but does not use. After QEMM has allocated memory for its data, High RAM and other resources, the remaining memory on the system becomes QEMM's memory pool. Because allocations from QEMM's memory pool are based on the allocation practices of expanded memory, which is allocated in 16K chunks, any memory that cannot be allocated as part of a contiguous 16K chunk is reported as unassigned. Some of the QEMM parameters (RAM and ROM, primarily) will increase or decrease the amount of unassigned memory.

Unmappable Pages - On the Manifest Expanded Pages screen, unmappable pages are 16K segments of address space (aligned on 16K boundaries) into which expanded memory cannot be mapped for one of several reasons: because the segment is in use by an adapter, is filled with High RAM, is excluded on the QEMM386.SYS line in the CONFIG.SYS file, or has already been filled with expanded memory by some program.

Upper Memory Area - The area between the end of conventional memory (typically 640K) and 1024K, used by system hardware (such as the system ROM, video adapter, etc.) and by QEMM to create High RAM and an EMS page frame.

Upper Memory Block - RAM located above 640K and below the one megabyte boundary, accessed through XMS calls. Upper memory blocks are used by real-mode programs to move relatively small amounts of code and data out of conventional memory. With QEMM as memory manager, the same memory can be used as High RAM or as an upper memory block, depending on how it is requested.

V

VCPI - Virtual Control Program Interface. See EMS in **Appendix C**.

VDS - Virtual DMA Services. See **Appendix C**.

Video Adapter - The hardware and ROM programs that control the video display. Some standard types of video display are monochrome, Hercules, CGA, EGA, and VGA; more advanced types of video display are now common.

VIDRAM - A QEMM program that can use parts of the address space normally reserved for video memory, creating as much as 96K additional conventional memory for DOS text programs or 8514 graphics programs.

Virtual-8086 Mode - A mode of 80386 (and higher) processors that allows extended memory to be mapped into the first megabyte. Real-mode DOS programs can run without modification in this mode. Virtual-8086 mode allows QEMM to support EMS 4.0 without additional special hardware. See also Protected Mode and Real Mode.

Virtual Control Program Interface - See VCPI in **Appendix C**.

Virtual DMA Services - See VDS in **Appendix C**.

Virtual Memory - A feature of 80386 (and higher) processors that lets a memory manager or control program swap the contents of memory to the hard disk in 4K chunks when additional memory is needed, then restore the contents when the processor is about to access them. Virtual memory slows the system to some extent but makes it possible to run large or multiple programs with relatively little RAM. QEMM's QDPMI program makes use of virtual memory techniques.

X

XBDA - Extended BIOS Data Area.

XMS - Extended Memory Specification.

NOTES:

Index

C

D

E

F

G

H

M

N

O

P

Q

T

U

V

W

X

Quarterdeck Expanded Memory Manager (QEMM) is a memory manager produced by Quarterdeck Office Systems in the late 1980s through late 1990s. It was the most popular third-party memory manager for the MS-DOS and other DOS operating systems.